Cycle de vie sémantique de conception de systèmes de stockage et manipulation de données

Selma Khouri

ESI : Ecole nationale Supérieure d'Informatique (Algérie)
ISAE-ENSMA : Ecole Nationale Supérieure de Mécanique et d'Aérotechnique (France)

THESE

pour l'obtention du Grade de

DOCTEUR DE L'ESI & DE L'ISAE-ENSMA

Écoles Doctorales :
Sciences et Technologies de l'Information et de la Communication (STIC, ESI)
Sciences et Ingénierie pour l'Information, Mathématiques (S2IM, ISAE-ENSMA)
Secteur de Recherche : INFORMATIQUE

Présentée par :

Selma KHOURI

Cycle de Vie Sémantique de Conception de Systèmes de Stockage et Manipulation de Données

Soutenue le 13 Octobre 2013

JURY

Président	**Djamel Eddine ZEGOUR**	Professeur, ESI, Algérie
Rapporteurs	**Dominique LAURENT**	Professeur, Université de Cergy Pontoise, France
	Hafida BOUARFA-ABED	Professeur, Université de Blida, Algérie
Examinateurs	**Yamine AÏT AMEUR**	Professeur, ENSEEIHT/IRIT, France
	Patrick GIRARD	Professeur, Université de Poitiers, France
	Reda Abdessamad GHOMARI	Maître de Conférences A, HDR, ESI, Algérie
Directeurs	**Ladjel BELLATRECHE**	Professeur, ISAE/ENSMA, France
	Thouraya BOUABANA-TEBIBEL	Professeur, ESI, Algérie

Remerciements

Un grand merci à :

- **Ladjel BELLATRECHE**, mon directeur de thèse, pour m'avoir guidé tout au long de mon travail. Je le remercie pour sa disponibilité, pour ses précieuses orientations, pour sa passion pour la recherche et pour son soutien aussi bien sur le plan humain que scientifique.

- **Thouraya BOUABANA-TEBIBEL**, ma directrice de thèse, pour m'avoir conseillé tout au long de mon travail.

- Au corps administratif de l'ESI, particulièrement **Nacera CHERID** pour son aptitude à l'écoute et **KHELIFATI Si Larabi** pour sa bienveillance.

- **Emmanuel GROLLEAU** pour avoir bien voulu m'accueillir au sein du laboratoire LIAS qu'il dirige.

- **Stéphane JEAN** et **Michaël BARON**, pour leur aide et leur contribution à la rédaction de certains articles.

- Tous les membres du LIAS et particulièrement **Selma BOUARAR**, **Amira KERKAD**, **Ilyes BOUKHARI**, **Ahcene BOUKORCA**, **Thomas LACHAUME**, **Yassine OUHAMMOU**, **Chadlia CHAKROUN** et **Youness BAZHAR**.

- **Djamel Eddine ZEGOUR** pour m'avoir fait l'honneur d'être président du jury.

- **Hafida BOUARFA-ABED**, **Abdessamed Réda GHOMARI**, **Dominique LAURENT**, **Yamine AIT AMEUR** et **Patrick GIRARD** pour avoir accepté d'être membres du jury.

- Mes amies et particulièrement **Zakia**, **Yassmine**, **Amel**, **Lynda**, **Soumia**, **Fatima**, **Lydia**, **Sana**, **Rym** et **Touka**.

- Enfin et surtout à **ma famille** et particulièrement à **mes parents** pour leur soutien sans faille. Rien n'aurait été possible sans eux.

A tous ceux qui me sont chers...

Table des matières

Table des figures

Introduction Générale

Contexte

L'industrie des *systèmes de stockage des données* (les bases de données ou les entrepôts de données) représente actuellement un marché très fructueux générant plusieurs milliards de dollars par an ces dernières années. Ces systèmes permettent de gérer un des capitaux les plus importants de toute organisation ou entreprise : *ses données*. Le développement des systèmes de stockage des données passe par un cycle de conception qui comprend les cinq principales phases suivantes [142] : (1) la définition des besoins des utilisateurs, (2) la modélisation conceptuelle, (3) la modélisation logique, (4) la modélisation physique, (5) l'implémentation de la BD et son tuning.

L'établissement de ce cycle de conception a connu plusieurs étapes d'évolutions avant d'être adopté par la communauté des bases de données (BD). Cette évolution est liée aux propositions de différents *modèles de données* développés afin de gérer au mieux les données stockées. Les premiers modèles de données proposés à la fin des années 60, sont les modèles de données *physiques*. Deux principaux modèles ont connu un certain succès et ont été utilisés dans d'importants systèmes industriels : le modèle *hiérarchique* utilisé dans le système IBM IMS, et le modèle *en réseau* utilisé dans le système Codasyl (Conference on Data Systems Languages) [49]. Ces deux modèles ont marqué une évolution importante dans le domaine des BD, dans la mesure où ils ont représenté les premiers modèles permettant de gérer et d'interroger l'ensemble de données d'une BD. Le modèle Codasyl a valu à son auteur *Charles Bachman*, le prix ACM Turing en 1973. Cependant, comme toute première proposition, ces modèles présentaient de nombreux inconvénients. Le plus important est la difficulté de maintenance et d'évolution de la BD, dûe à la *faible indépendance physique* des données.

Afin de remédier à cet inconvénient, le modèle *relationnel* a été proposé par *Edgar Codd* [50]. L'indépendance physique des données signifie la capacité de modifier le schéma interne de la BD sans avoir à modifier ni son schéma conceptuel, ni les schémas externes des utilisateurs. Le modèle relationnel, a ainsi été conçu pour présenter une *abstraction mathématique de ce qui est implémenté* dans la BD. Le modèle relationnel est basé sur une représentation formelle. Il propose de stocker les données dans un modèle organisé en des structures simples et flexibles (des tables) pouvant représenter tout type d'information. Ce modèle marque une nouvelle évolution importante, sanctionnée également par un prix Turing en 1981. Le modèle relationnel a été implémenté dans d'importants projets académiques et industriels comme le projet *SystemR*, le projet *Ingres* et le projet *Gamma* [76].

Malgré l'engouement suscité par le modèle relationnel, les éditeurs de logiciels et les utili-

sateurs en entreprises ont constaté que les modèles existants ne répondaient pas suffisamment à leurs attentes. Ils réclamaient des modèles de données incorporant davantage de sémantique représentée par les règles de gestion de leur entreprise, et souhaitaient avoir une méthodologie unifiée pour la conception de leurs BD [45]. C'est dans ce contexte que *Peter Chen* proposa en 1976, son modèle *entités/associations* (E/A). Ce dernier offre une vision de la BD, sous forme d'un modèle conceptuel, plus proche de l'utilisateur humain. Il apporte un nouveau niveau d'abstraction qui fournit une *indépendance logique des données*. L'indépendance logique signifie la capacité de modifier le schéma conceptuel de la BD sans avoir à modifier ses schémas externes ou ses programmes d'application. Ces trois principales propositions apportées par les trois modèles (codasyl, relationnel et E/A) ont permis l'établissement d'un premier cycle de conception des BD. Ce cycle a été matérialisé dans l'architecture ANSI/SPARC qui distingue les trois niveaux d'abstraction pour la conception des BD : le niveau conceptuel, le niveau externe et le niveau interne. Nous appelons cette première génération d'évolutions "évolution verticale".

Une seconde génération d'évolution, a permis de détailler chaque phase de conception par un ensemble d'étapes à suivre pour mener à bien le processus de conception de la BD. La phase de définition des besoins permet d'identifier l'ensemble des données et des traitements de la BD à développer. La phase de modélisation conceptuelle fournit un modèle conceptuel facile à comprendre et à valider par les utilisateurs. La phase de modélisation logique fournit un modèle logique représentant l'organisation des données dans un modèle qui peut être implémenté. La phase de modélisation physique fournit un modèle physique des données correspondant au SGBD choisi pour l'implémentation de la BD. Cette évolution "interne" à chaque phase, a permis la stabilité du cycle de conception.

Une troisième génération d'évolution que nous appelons "évolution horizontale", a permis d'étendre les modèles de données proposés au sein de chaque phase de conception. Le cycle de conception a ainsi connu une multiplication des modèles conceptuels, logiques, physiques et des architectures de déploiement de la BD. Par exemple, de nouveaux modèles sémantiques, orientés objets et les modèles ontologiques ont montré leur contribution au sein de la phase de modélisation conceptuelle. De nouveaux modèles logiques sont également apparus avec le développement des bases de données orientées objets, relationnelles objet, spatiales, XML, ontologiques, etc. Avec l'évolution de la technologie des bases de données et le développement des réseaux et des systèmes de stockage et de traitement, la phase d'implémentation a également connu une diversification des architectures de déploiement de la BD. La phase d'implémentation doit inclure les architectures matérielles sur lesquelles les données sont déployées comme les BD distribuées, les BD parallèles, les grilles de calcul, les BD sur flash, BD dans le cloud, etc.

La stabilité des phases du cycle de conception a permis le développement de plusieurs applications autour des systèmes de gestion des données. Ces données ont dû être analysées afin d'en extraire de nouvelles connaissances utiles pour la gestion stratégique de l'organisation. Une nouvelle architecture matérialisant les données a été proposée sous forme d'un *entrepôt de données* (ED). Les ED permettent d'intégrer les données issues de différentes sources, souvent hétérogènes, au sein d'une base commune. Les ED présentent plusieurs similarités avec les BD, dans le sens où les deux architectures permettent le stockage et la consolidation de données issues de diverses sources (fichiers, BD, etc). Les BD et les ED diffèrent principalement dans leur usage. En effet, alors que les BD supportent la gestion de l'activité quotidienne de l'entreprise, les ED

permettent l'analyse des données de l'entreprise.

La similarité entre les ED et les BD est également constatée dans leur cycle de conception. Le cycle de conception des ED a connu deux principales évolutions. La première génération des projets d'ED s'est concentrée sur les trois phases de modélisation logique, la phase ETL et la modélisation physique. Ceci s'explique par le fait que les projets d'entreposage sont issus originellement de l'industrie qui s'intéresse davantage à l'amélioration et à l'optimisation des performances du système décisionnel final. Ces tâches sont traitées lors de la conception logique et surtout physique de l'ED. La phase ETL a été introduite suite à la phase de conception logique, elle consiste à extraire les données des sources afin de peupler le schéma cible de l'ED.

Ce cycle de conception des ED a ensuite été étendu pour intégrer deux nouvelles phases : la phase de définition des besoins et la phase de modélisation conceptuelle. Divers travaux et études ont montré l'importance de ces deux phases pour assurer le succès des projets d'entreposage. Ce cycle complété ressemble ainsi au cycle de conception des BD. Nous remarquons que l'apparition des projets d'ED et leur évolution n'a fait que ***consolider*** ce cycle de conception.

L'importance de chaque phase du cycle de conception établi a fait naître plusieurs communautés de recherche. Ces communautés sont reconnues autour de conférences internationales portant sur les problèmes liés à chaque phase (*Requirement Engineering*, *INFORSID* pour la collecte des besoins), (*ER*, *INFORSID* pour la modélisation conceptuelle), (*VLDB*, *SIGMOD*, *ICDE*, *BDA* liées principalement à la phase logique et physique). Deux principaux constats sont faits suite à l'analyse du cycle de conception, et qui sont à l'origine de nos contributions.

Nous soutenons la thèse que nos deux principales contributions s'appliquent aussi bien pour les BD que pour les ED. Nous avons choisi de concentrer notre étude sur les applications d'ED pour deux principales raisons : (1) les ED sont des systèmes complexes qui ont contribué à l'évolution du cycle de conception, par l'apparition de nouveaux modèles de données (conceptuel, logique et physique). (2) Certaines phases, comme la phase de modélisation physique et la phase d'implémentation et de tuning, ont connu davantage d'importance avec l'apparition des applications décisionnelles basées sur les ED, car ces dernières manipulent un grand volume de données et exigent un temps de réponse raisonnable pour satisfaire les exigences des décideurs.

Problématique et contributions

Le premier constat fait état du modèle de données matérialisé au sein de l'ED. De l'ensemble des phases de conception, seul les modèles logique et physique de données sont représentés au sein de l'ED. Ces modèles présentent des modèles ayant subi diverses transformations, et renferment très souvent des décisions d'optimisation et d'implémentation. Par conséquent, ils ne représentent pas fidèlement le modèle externe reflétant les vues des utilisateurs. Certaines BD à base ontologique ont été proposées récemment. Elles permettent de stocker, en plus du modèle logique de données, le modèle conceptuel de la base sous forme d'une ontologie locale. La présence du modèle conceptuel permet de fournir une sémantique aux données de la base, et facilite ainsi l'interrogation de la BD et sa future intégration. Ces modèles (conceptuel, logique et physique) représentent uniquement les *données* à différents niveaux d'abstraction. Les *traitements* que doit effectuer l'ED, et qui sont identifiés lors de la première phase de définition des besoins, ne sont aucunement représentés dans la structure finale de l'ED. Ces traitements sont nécessaires tout au

long du cycle de vie de l'ED pour ses tâches de conception, d'optimisation, de personnalisation et de maintenance.

En effet, l'analyse de l'évolution du cycle de conception des systèmes de stockage des données nous a montré la multiplicité des modèles de données conceptuels, logiques, et physiques et la diversification des plateformes de déploiement. Cette diversification offre plusieurs choix de conception, mais nécessite de devoir effectuer des simulations du comportement de l'ED à différentes phases de conception, afin d'opter pour la meilleure représentation possible. Par exemple, les négociateurs du client du système et les représentants des éditeurs logiciels doivent répondre aux choix pertinents des modèles et des architectures de l'ED. En fonction des données et des traitements que devra effectuer l'ED, le concepteur choisit les modèles de données offrant la meilleure représentation pour l'ED. Si l'entreprise possède un administrateur, ce dernier à l'aide de simulations (avec des bancs d'essai) peut donner des recommandations pour acquérir la technologie susceptible de gérer l'ED. Par conséquent, la multiplicité des modèles de stockage et des architectures supportant les systèmes d'ED exige une communication étroite entre les acteurs des différentes phases du cycle de vie de l'ED. Pour pouvoir effectuer de telles simulations, l'unique ressource disponible permettant de fournir les données de l'ED et aussi ses traitements consiste en l'ensemble des *besoins des utilisateurs*.

Première proposition : persistance des besoins des utilisateurs

Notre première proposition a consisté à revisiter le cycle de conception afin de faire persister les besoins des utilisateurs. Cette révision permet de répondre mieux aux dernières évolutions du cycle de conception, et permet de faire des choix plus pertinents par rapport aux différents modèles de données et architectures proposées. Nous avons concrétisé notre proposition en fournissant une méthode de conception dédiée aux ED, permettant de faire persister les besoins des utilisateurs dans la structure finale de l'ED. Récemment, les méthodes de conception des ED proposées dans la littérature ont montré l'intérêt de l'utilisation des ontologies dans le processus de conception. Les ontologies ont montré leur contribution, du fait de leur similarité avec les modèles conceptuels et du fait qu'elles règlent les problèmes d'hétérogénéité syntaxiques et sémantiques lors de l'analyse des sources de données. Ces méthodes supposent l'existence d'une ontologie de domaine partagée et référencée par l'ensemble des sources qui alimenteront l'ED.

Dans notre méthode, nous étendons l'utilisation de l'ontologie pour l'analyse et la spécification des besoins des utilisateurs. Les méthodes existantes considèrent les besoins des utilisateurs comme le cahier des charges de l'application décisionnelle à développer. En poussant cette réflexion, l'ontologie peut être vue comme le cahier des charges couvrant l'ensemble des besoins du domaine, qui est défini de manière formelle et consensuelle. La méthode que nous proposons permet de fournir le schéma de l'ED défini dans une architecture matérialisant le modèle de données ainsi que le modèle de besoins. Une fois le schéma de l'ED développé, son peuplement par les données des sources s'effectue via la phase de conception ETL.

Deuxième proposition : un ETL conceptuel pour un déploiement à la carte

Le deuxième constat a été fait suite à l'analyse du positionnement de la phase ETL dans le cycle de conception des ED. Dans les premiers projets de BD, La phase ETL était déjà présente

et consistait en un traitement logiciel de données qui filtre, calcule de nouvelles valeurs, et remplit un autre espace de stockage que l'original. Après l'apparition des ED, l'ETL était là, cependant toujours cachée entre les lignes. Ce n'est qu'au cours des années 2000, que l'ETL a obtenu une existence séparée en devenant une phase importante pour l'intégration des sources, étant donné sa consommation en terme d'argent, de temps et de main d'œuvre [201]. La phase modélisation conceptuelle n'étant pas encore considérée, la phase ETL a été introduite après la phase de modélisation logique. Cette dernière inclut des choix d'implémentation qui influencent le schéma de l'ED et lui imposent un déploiement unique. Ceci est dû au fait que le schéma de l'ED est figé dès le début du processus de conception. Ce schéma suit la représentation des schémas des sources (généralement relationnelle). La mise en correspondance entre les schémas des sources de données et le schéma cible de l'ED s'effectue selon ce choix unique de représentation et se fait après la phase de modélisation logique. L'analyse de l'évolution du cycle de conception des ED nous a montré la diversification des modèles logiques, physiques et des plateformes de déploiement disponibles. La conception de l'ED doit nécessairement s'adapter à ce cycle de conception. Notre deuxième proposition a consisté à revisiter le cycle de conception en étudiant l'ordonnancement de la phase de conception ETL. Afin de fournir une flexibilité dans le déploiement de l'ED, nous avons proposé de redéfinir la phase de conception ETL au niveau conceptuel afin de la libérer de toute contrainte d'implémentation. Ceci permet un déploiement "à la carte" de l'ED selon les besoins des utilisateurs et selon le choix du concepteur ou de l'administrateur. Notre analyse de la littérature relative à la conception des ED, nous a fait constater que les travaux proposés étudient la conception du schéma de l'ED et son alimentation par le processus ETL de manière isolée. Parmi les travaux s'intéressant à la conception ETL, nous remarquons que des méthodes récentes définissent certains éléments de la phase ETL au niveau conceptuel.

Ces travaux considèrent l'ED comme un *système d'intégration* constitué d'un schéma global, des schémas des sources et des correspondances entre le schéma global et les schémas des sources. Le processus d'intégration s'effectue via un algorithme ETL, et permet de matérialiser les données des sources dans l'ED. La principale difficulté rencontrée lors la mise en œuvre d'un système d'intégration de données consiste en l'hétérogénéité syntaxique et sémantique des sources. L'hétérogénéité syntaxique provient du fait que les sources peuvent avoir différentes structures et formats pour stocker leurs données. L'hétérogénéité sémantique provient des différentes interprétations des objets du monde réel. Pour traiter l'hétérogénéité sémantique, considérée comme la principale difficulté, l'utilisation des *ontologies* est apparue comme l'approche la plus prometteuse, et présente la clé de l'automatisation du processus d'intégration.

Dans ce type d'approche, la sémantique de chaque source participant au processus d'intégration est explicitée par une ontologie locale. Chaque ontologie locale référence une ontologie partagée supposée existante. Cette hypothèse est raisonnable dans plusieurs domaines d'application, où les ontologies sont largement développées : dans le domaine de l'ingénierie, où les catalogues de composants électroniques sont souvent décrits par des ontologies normalisées par ISO (PLIB : ISO 132584 "parts Library"), dans le domaine de la médecine, nous trouvons l'ontologie UMLS (Unified Medical Language System [1]), la biologie [2], les applications de l'intelligence économique, etc. Le stockage des ontologies dans une base de données a engendré la notion de

1. http ://www.nlm.nih.gov/research/umls/
2. http ://www.iplantcollaborative.org/

bases de données à base ontologique (BDBO). Plusieurs systèmes académiques et industriels ont proposé des solutions efficaces de BDBO pour stocker et gérer les ontologies et les données associées. Les BDBO sont par conséquent utilisées dans les entreprises et deviennent de ce fait des sources candidates pour les projets d'ED. Cependant, aucune proposition n'a été faite pour la définition d'un ED à partir de BDBO.

Nous avons concrétisé notre deuxième proposition en fournissant une méthode de conception complétant la méthode issue de notre première proposition pour la conception du schéma de l'ED, et définissant la phase ETL à l'étape conceptuelle plus précisément à un niveau ontologique. Nous fournissons ainsi une approche ontologique couvrant l'ensemble des phases du cycle de conception de l'ED. Cette approche s'appuie sur la définition d'un ensemble de besoins des utilisateurs, et la définition d'un Framework d'intégration formalisant les sources de données, le schéma global et les correspondances entre l'ensemble de ces schémas. Nous avons défini un algorithme ETL reposant sur des opérateurs ETL conceptuels, pour l'intégration des données des sources. Nous instancions ensuite le Framework proposé par des sources de données de type BDBO. Nous appliquons finalement notre méthode pour la conception du schéma de l'ED et son alimentation par les données issues des BDBO.

Les principales contributions

Nous faisons remarquer que nos deux propositions ont émergé de l'analyse du cycle de conception des BD/ED et de son évolution. Ces contributions ont été proposées dans une logique de *consolidation* du cycle de conception. Elles permettent d'offrir une conception plus flexible et plus adaptée aux évolutions récentes et aux diverses propositions des modèles de données et des architectures d'implémentation.

Nous résumons les principales contributions de chaque proposition comme suit :

1. *Etats de l'art* : présentation d'une synthèse détaillée ainsi qu'une classification des principaux travaux de conception du schéma de l'ED et des travaux de conception ETL. Ces travaux sont analysés selon différents critères de comparaison.
2. *Persistance des besoins pour un système d'ED manipulant données et traitements* : pour notre première proposition, trois principales contributions sont présentées :
   - Proposition d'une méthode de conception du schéma de l'ED, spécifiant les sources de données et les besoins des utilisateurs au niveau ontologique. Nos articles présentant cette contribution sont les suivants [106, 99, 171].
   - Proposition d'une architecture d'ED faisant cohabiter les deux principaux modèles de l'ED : le modèle de données et le modèle de besoins [103, 104, 98, 105].
   - Prototypage et tests : un prototype d'outil implémentant les étapes de la méthode est proposé [97, 96]. Plusieurs expérimentations ont été menées utilisant le banc d'essai Star Schema Benchmark (SSB). Ces expérimentations permettent de simuler différentes tâches d'optimisation de l'ED à partir des besoins des utilisateurs [19, 102].
3. *ETL au niveau ontologique pour un déploiement à la carte* : pour notre seconde proposition, quatre principales contributions sont présentées :
   - Proposition d'un Framework d'intégration générique <G,S,M> pour l'intégration des sources au niveau conceptuel [100].

– Proposition d'une méthode de conception et d'alimentation d'un ED définissant la phase ETL dès la phase conceptuelle, plus précisément au niveau ontologique [23].
– Application de la méthode de conception proposée par l'instanciation du Framework d'intégration par des sources de type BDBO [18].
– Prototypage et tests : un prototype d'outil implémentant l'ensemble des étapes de la méthode est fourni [18]. Des expérimentations ont été conduites, utilisant le banc d'essai LUBM, et permettent d'évaluer les performances de la méthode proposée [22].

Organisation de la thèse

Ce manuscrit de thèse comprend deux principales parties.

Partie états de l'art

La première partie présente notre état de l'art, et s'articule autour de deux chapitres. Le premier chapitre décrit les principales évolutions du cycle de conception des BD, ensuite des ED. Les principales notions relatives aux ED sont présentées dans ce chapitre.

Le deuxième chapitre présente les principaux travaux proposés dans la littérature relative aux projets d'ED. Nous commençons par analyser les travaux permettant de concevoir le schéma de l'ED. Nous analysons ensuite les travaux de conception ETL pour l'alimentation de l'ED. Une classification de l'ensemble des travaux cités selon un ensemble de critères retenus est présentée. Le positionnement de notre approche par rapport aux travaux de l'état de l'art est ensuite analysé.

Partie propositions

La deuxième partie présente nos propositions et contributions, et s'articule autour de deux chapitres.

Le troisième chapitre présente notre première proposition. Les fondements théoriques sur lesquels repose notre méthode sont définis. Ces fondements présentent les principales notions relatives aux ontologies et aux BDBO. La méthode proposée est ensuite décrite, où nous présentons une formalisation des entrées et sortie de la méthode et une description de l'ensemble des étapes à suivre. La dernière partie du chapitre décrit les expérimentations effectuées pour valider notre méthode.

Le quatrième chapitre présente notre deuxième proposition. Les fondements théoriques sur lesquels repose notre méthode sont définis. Ces fondements présentent les principales notions relatives aux systèmes d'intégration de données. La méthode proposée est ensuite décrite, où nous présentons le Framework d'intégration et l'algorithme ETL. La méthode de conception couvrant l'ensemble des phases de conception de l'ED est présentée. La méthode est ensuite appliquée et validée sur un ensemble de BDBO. La dernière partie du chapitre présente l'outil implémentant la méthode et décrit les expérimentations effectuées.

Le cinquième chapitre présente les conclusions générales de ce travail et esquisse diverses perspectives.

Partie annexes

Ce manuscrit comporte deux annexes. L'annexe A décrit les besoins et requêtes du banc d'essai Star Schema Benchmark utilisé pour valider la première proposition. L'annexe B fournit les détails des nos publications et leurs principales contributions.

Publications

La liste suivante présente les publications concernant le travail de cette thèse.

Liste des articles relatifs à la première proposition

1. Selma KHOURI, Ladjel BELLATRECHE, A Methodology and Tool for Conceptual Designing a Data Warehouse from Ontology-based Sources, ACM Thirteenth International Workshop On Data Warehousing and OLAP (DOLAP 2010), Toronto, Canada, edited by ACM, October, 2010, pp. 19-23. [106].
2. Selma KHOURI, Ladjel BELLATRECHE, OntoStar : Outil de conception multidimensionnelle d'un entrepôt de données à base ontologique, 26èmes journées Bases de Données Avancées (BDA'2010), October, 2010. [96].
3. Selma KHOURI, Ladjel BELLATRECHE, DWOBS : Data Warehouse Design from Ontology-based Sources, in 16th International Conference on Database Systems for Advanced Applications (DASFAA' 2011), April, 2011, pp. 438-441. [97].
4. Selma KHOURI, Ladjel BELLATRECHE, Osons tout Persister dans un Entrepôt : Données et Modèles, 7èmes Journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'11), edited by RNTI, 2011. [98].
5. Selma KHOURI, Ladjel BELLATRECHE, "Valorisation des besoins des utilisateurs dans le processus de conception et d'exploitation des entrepôts de données, 4èmes Journées Francophones sur les Ontologies (JFO'2011), 2011. [99].
6. Selma KHOURI, Ladjel BELLATRECHE, Patrick Marcel, Embedding User's Requirements in Ontology-based Data Warehouses, International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE), October, 2011. [103].
7. Selma KHOURI, Ladjel BELLATRECHE, Patrick Marcel, Towards a Method for Persisting Requirements and Conceptual Models in Data Warehousing Context, 27èmes journées de Bases de Données Avancées (BDA'2011), 2011. [171].
8. Ladjel BELLATRECHE, Selma KHOURI, Ilyes BOUKHARI, Rima BOUCHAKRI, Using Ontologies and Requirements for Constructing and Optimizing Data Warehouses, Proceedings of the 30th IEEE International Convention MIPRO, Opatija, Croatia, May, 2012. [19].
9. Selma KHOURI, Ladjel BELLATRECHE, Ilyes BOUKHARI, Selma Bouarar, More Investment in Conceptual Designers : Think about it !, 15th IEEE International Conference on Computational Science and Engineering., 2012. [102].

La première proposition de notre étude a fait l'objet de deux publications. La première publication dans une revue francophone [104] est issue de l'article [105]. La deuxième publication a été publiée dans une revue internationale.

1. Selma KHOURI, Ladjel BELLATRECHE, Patrick Marcel, Une démarche de conception d'un Entrepôt Sémantique Matérialisant les Données et les Besoins, Ingénierie des Systèmes d'Information, 2012.
2. Selma KHOURI, Ilyes BOUKHARI, Ladjel BELLATRECHE, Stéphane JEAN, Eric SARDET, Mickael BARON, Ontology-based structured web data warehouses for sustainable interoperability : requirement modeling, design methodology and tool, Computers in Industry Journal, Elsevier (Factor Impact : 1.529), 2012.

Liste des articles relatifs à la deuxième proposition

1. Selma KHOURI, Ladjel BELLATRECHE, Nabila Berkani, Generic Conceptual Framework for Handling Schema Diversity in Data Integration : Applications to Semantic Databases, 16th East-European Conference on Advances in Databases and Information Systems (ADBIS), edited by Springer Verlag, 2012. [100]
2. Selma KHOURI, Nabila Berkani, Ladjel BELLATRECHE, Generic Methodology for Semantic DataWarehouse Design : From Schema Definition to ETL, To appears in 4-th International Conference on. Intelligent Networking and Collaborative Systems (INCoS), edited by IEEE, 2012. [23].
3. Selma KHOURI, Ladjel BELLATRECHE, Nabila Berkani, MODETL : A complete MODeling and ETL method for designing Data Warehouses from Semantic Databases, International Conference on Management of Data (COMAD), 2012. [101].
4. Ladjel BELLATRECHE, Selma KHOURI, Nabila Berkani, Semantic Data Warehouse Design : From ETL to Deployment à la Carte, 18th International Conference on Database Systems for Advanced Application (DASFAA 2013), 2013. [18].

La deuxième proposition de cette thèse a fait l'objet d'une publication internationale.

1. Nabila Berkani, Ladjel BELLATRECHE, Selma KHOURI, Towards a Conceptualization of ETL and Physical Storage of Semantic Data Warehouses as a Service, Cluster Computing Journal (CCJ 2013), 2013. [22].

Première partie

Etat de l'art

Chapitre 1

Cycle de conception des systèmes de stockage des données

Sommaire

1.1 Introduction

Les BD permettent de gérer un des capitaux les plus importants d'une organisation : *les données* que l'organisation traite et manipule. Dans les organisations manipulant d'importantes quantités de données, les BD font partie intégrante des systèmes d'informations (SI) et représentent l'une de leurs plus importantes composantes. Le cycle de vie d'un SI comprend les phases suivantes [55] (figure 1.1) :

- l'analyse de la faisabilité du système,
- la collecte et l'analyse des besoins,
- la conception du système,
- l'implémentation,
- la validation et les tests,
- le déploiement et la maintenance du système.

Plusieurs méthodologies de conception ont été proposées pour supporter ce cycle. Nous citons les méthodes *cartésiennes* (comme la méthode SADT pour Structured Analysis and Design Technique) qui sont basées sur la décomposition hiérarchiques des processus et des flux de données, les méthodes *systémiques* (comme la méthode Merise) centrées sur la modélisation des données, ou les méthodes *objet* proposant d'intégrer l'aspect dynamique du système à son aspect structurel [161].

Le cycle de vie d'une BD, appelé "cycle de vie micro", par analogie au cycle de vie des systèmes d'informations appelé "cycle de vie macro", comprend les phases suivantes : la définition du système, la conception et l'implémentation de la BD, le chargement des données et leur conversion (en cas de migration d'une ancienne base existante), la conversion de l'application, la validation et les tests, l'opérationnalisation de la BD et sa maintenance. Au cours de ce chapitre, nous nous focalisons sur les phases centrales de conception et d'implémentation de la base, constituant "le cycle de conception de la BD". Ce cycle de conception comporte les cinq phases suivantes [142] (figure 1.2) :

- la définition des besoins des utilisateurs,
- la modélisation conceptuelle,
- la modélisation logique,
- la modélisation physique et
- l'implémentation de la BD et son tuning.

Pour aboutir à ce consensus concernant ces étapes du cycle de conception, le domaine de conception des BD a connu plusieurs évolutions. Les principales contributions de notre présente étude ont consisté à revisiter ce cycle de conception afin de répondre au mieux aux dernières évolutions. Répondant à la citation énonçant que "*la connaissance s'acquiert par le cumul des expériences*", nous exposons dans ce qui suit les évolutions les plus marquantes ayant mené à l'établissement de ce cycle de conception et son adoption par la communauté de BD, en prenant comme date source l'apparition des premiers modèles de données. Nous montrons dans la deuxième section que ce même cycle a été adopté et consolidé pour les systèmes d'entrepôt de données. Cette revue de littérature nous permettra de positionner nos contributions, qui se situent dans la même lignée de consolidation du cycle de conception des BD.

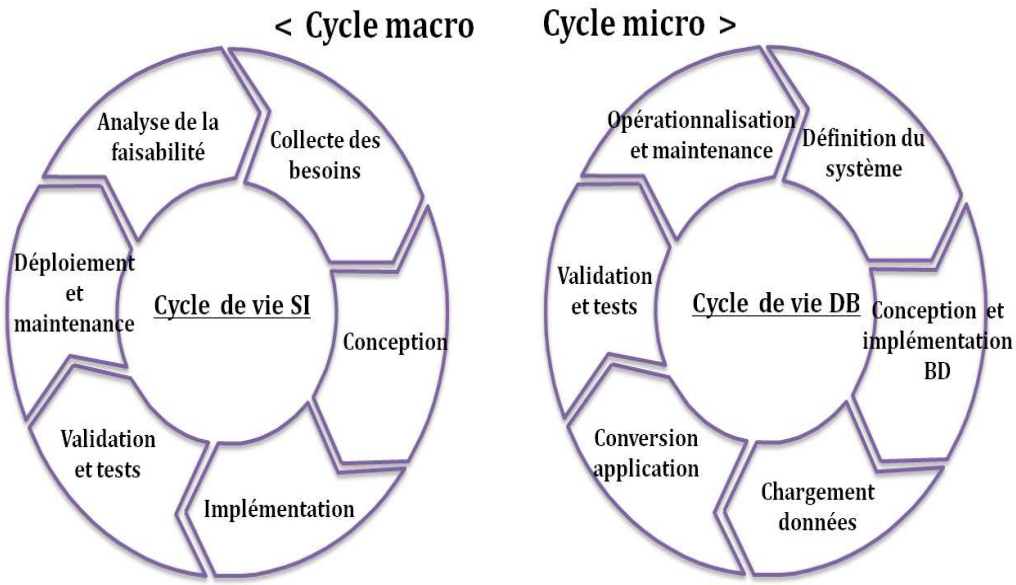

FIGURE 1.1 – Cycle de vie d'un système d'information et d'une BD

1.2 Cycle de conception des bases de données

Nous synthétisons les principales étapes ayant mené au consensus concernant le cycle de conception des BD en trois principales évolutions (figure 1.3) : une évolution *verticale* où les principales phases de modélisation (analyse des besoins, phase de modélisation conceptuelle, logique et physique) ont été ajoutées de manière évolutive au cycle de conception. Une deuxième évolution *interne* qui a permis de détailler les étapes à suivre pour chacune des phases du cycle. Une troisième évolution *horizontale* qui a permis d'enrichir chaque phase par l'apparition de nouveaux modèles de données. Nous présentons dans ce qui suit ces différentes évolutions.

1.2.1 Evolution verticale du cycle de conception

Les modèles de données ont été proposés afin de représenter la sémantique permettant rattacher à chaque information du monde perçu modélisé qui y est représentée, un sens. Les évolutions du cycle de conception des BD que nous présentons sont principalement liées aux différentes propositions des modèles de données.

1.2.1.1 Les modèles physiques

Après l'apparition de la technologie des BD dans les années 60 comme successeurs aux systèmes de fichiers, certains modèles de données ont été proposés pour gérer les données au sein d'un système de gestion de base de données (SGBD). Le principal objectif des SGBD est d'assurer une indépendance (la plus grande possible) entre les données d'une part et les programmes qui exploitent la BD d'autre part afin de gagner en souplesse.

Deux principaux modèles ont connu un certain succès et ont été utilisés dans des systèmes industriels : le modèle *hiérarchique* utilisé dans le système IBM IMS développé en 1968, et le

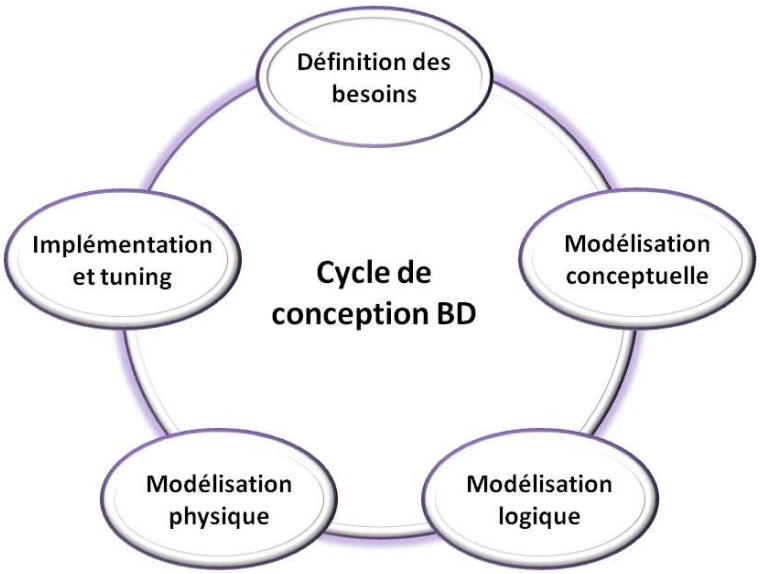

FIGURE 1.2 – Cycle de conception d'une BD

modèle *réseau* proposé par le groupe de travail DBTG (Data Base Task Group) du CODASYL (COnference on DAta SYstems Languages) en 1969 dans sa première version [49]. Ces systèmes ont été développés à la fin des années 60, leurs deux modèles de données ont été définis par la suite, par abstraction des modèles implémentés dans ces systèmes [49]. Le concept de "modèle de donnée" a émergé avec l'apparition de ces modèles, même si le terme *modèle de données* a été introduit et défini plus tard par *Edgar Codd* pour son modèle de données relationnel.

Le premier modèle *hiérarchique* (figure 1.4) consiste à stocker les données dans des enregistrements comportant un ensemble de champs ayant chacun un type de données. Les enregistrements sont organisés sous forme d'un arbre tel que chaque nœud (enregistrement) possède un seul parent. Chaque arbre comprend une racine et des branches qui permettent l'accès aux différents niveaux de données. Le niveau d'une donnée mesure sa distance à la racine. La racine est située au niveau supérieur. Les autres données se situent au niveau des nœuds de l'arbre (on parle de parents et d'enfants). Les relations entre les enregistrements d'un niveau et du niveau immédiatement inférieur sont de type *un à plusieurs* (1,n).

Les SGBD hiérarchiques les plus utilisés sur le marché étaient ceux des systèmes IMS et SYSTEM-2000. Leurs BD consistaient alors en un ensemble d'instances de ces enregistrements hiérarchiques. Ce modèle en arbre possède deux principaux inconvénients [85] : (1) la redondance de l'information pouvant engendrer des inconsistances entre les données. (2) l'existence des données d'un enregistrement repose sur celle de son enregistrement parent.

Le modèle de données *réseau* (ou codasyl) a été proposé en 1969, suivi d'une autre version en 1973 complétant le modèle par une spécification d'un langage de manipulation des données. Le modèle réseau a servi de base pour la plupart des systèmes commerciaux au cours des années 1970 [76]. Le modèle réseau (figure 1.5) stocke les données dans une collection d'enregistrements organisés, non pas dans un arbre, mais dans un réseau (ou graphe) dont les nœuds sont les enregistrements. Ce modèle permet ainsi à une instance d'un enregistrement d'avoir plusieurs parents et non plus un parent unique. Le langage de manipulation des données Codasyl est

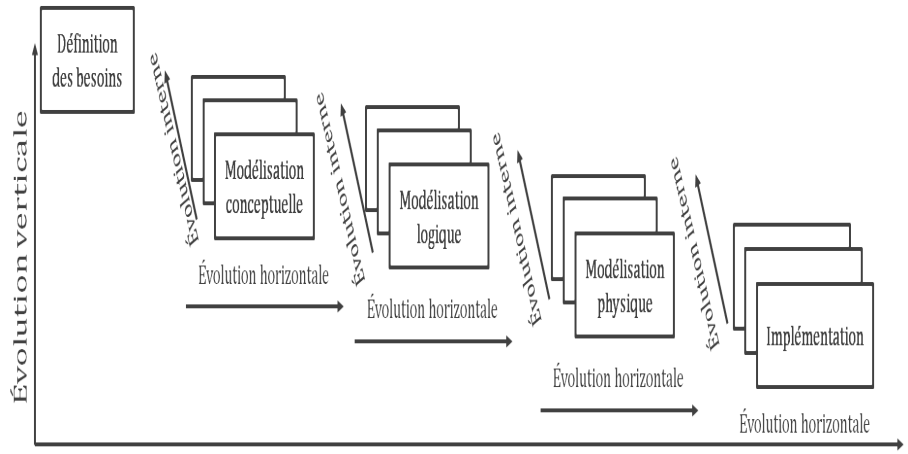

FIGURE 1.3 – Types d'évolution du cycle de conception des BD : horizontale, verticale, interne

un langage procédural permettant de récupérer et de manipuler un seul enregistrement à la fois. L'utilisateur accède à une donnée par un point d'entrée et navigue à travers le réseau à l'enregistrement contenant la donnée. Le modèle réseau est une évolution logique du modèle hiérarchique. Il offre une meilleure représentation des données et davantage de flexibilité que le modèle hiérarchique. Par rapport à ce dernier, le modèle réseau a permis l'élimination des redondances de données et la création de chemins d'accès multiples à une même donnée. Il a d'ailleurs valu à son auteur *Charles Bachman* d'avoir le prix ACM Turing en 1973. Cependant, ce modèle ne permet pas la prise en compte réelle de phénomènes plus complexes et ne résout pas tous les problèmes d'indépendance des structures de données et des traitements.

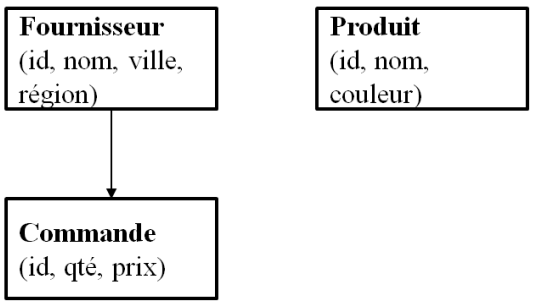

FIGURE 1.4 – Exemple d'une représentation hiérarchique

Ses principaux inconvénients sont les suivants :

- La difficulté d'installation et de gestion du système à cause du temps nécessaire pour l'organisation des données dans les deux niveaux (physique et logique) avant l'activation de la base.
- La difficulté de maintenance et d'évolution de la BD et des programmes.
- La difficulté d'accès aux données où les programmeurs étaient obligés de naviguer le long de chemins d'accès pour atteindre une donnée cible.
- La redondance des données qui nécessitait une révision simultanée de tous les programmes utilisant une structure ayant subi une modification [50].

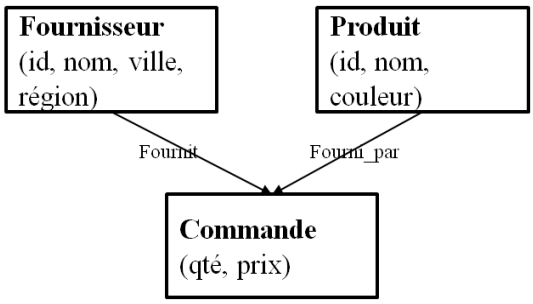

FIGURE 1.5 – Exemple d'une représentation en réseau

Les deux modèles hiérarchique et réseau comportent un inconvénient commun et majeur qui est la *faible indépendance physique et logique des données*. Ceci nécessitait de prévoir toutes les données et opérations nécessaires avant la conception de la BD, afin d'éviter de recoder d'éventuels changements ou mises à jour. Les modèles proposés étaient certes efficaces pour les requêtes et les opérations initiales pour lesquelles la BD a été conçue. Ils ne fournissaient cependant pas suffisamment de souplesse pour accéder aux enregistrements efficacement lorsque de nouvelles requêtes et transactions étaient identifiées. Ces transactions étaient souvent coûteuses en temps et en argent, car les programmes devaient être réécrits, testés et débogués [55].

1.2.1.2 Le modèle relationnel

L'indépendance physique des données était une nécessité car une application de BD n'est jamais écrite dès le premier jet, mais nécessite plusieurs mises à jour. Le modèle qui suivit ces deux premiers modèles, a été proposé avec pour principal objectif d'offrir une séparation entre les deux niveaux d'abstraction des données : le niveau logique et le niveau physique [85]. C'est ainsi qu'*Edgar Codd* a proposé en 1970 le modèle *relationnel* [48], présentant une abstraction mathématique de ce qui est implémenté dans la base. Le modèle relationnel propose de stocker les données dans un modèle organisé en des structures simples et flexibles (des tables) pouvant représenter tout type d'information. La gestion des données dans le modèle relationnel se fait par leur valeur et non par leur position souvent difficile à maintenir (comme dans le modèle réseau). Le modèle relationnel offre un accès aux données par un langage de manipulation permettant aux utilisateurs d'exprimer des opérations sur des blocs ou ensembles d'information à la fois (exploitant les concepts de la théorie des ensembles). Ce modèle présente de plus une représentation formelle et rigoureuse, basée sur l'algèbre relationnelle, pour l'organisation et la gestion de la BD. *Edgar Codd* définit ensuite son modèle de données par trois composantes [50]. Cette définition a été retenue pour les autres types de modèle de données :

- *Partie structurelle* : représente la collection de types de structures de données. La partie structurelle dans le modèle relationnel est représentée par les domaines, relations, attributs, tuples, clés candidates et primaires.
- *Partie manipulation* : collection d'opérateurs et de règles d'inférence qui peuvent être appliqués sur des instances valides de la BD pour leur restitution, leur modification ou leur gestion. La partie manipulation dans le modèle relationnel est représentée par les opérateurs algébriques (select, project, join, etc) qui transforment des relations en d'autres

relations.

– *Partie intégrité* : collection de règles d'intégrité qui définissent implicitement ou explicitement l'ensemble des états consistants de la base. La partie intégrité dans le modèle relationnel est représentée par les contraintes d'intégrité d'entités et d'intégrité référentielle.

L'ensemble de ces contributions ont valu à *Edgar Codd* de recevoir le prix Turing en 1981 [50]. Malgré ses avantages, le modèle relationnel n'a pas été accepté rapidement par la communauté et un grand débat eut lieu dans les années 70 dans les grandes conférences sur les BD (comme *SIGFIDET* et son successeur *SIGMOD*) opposant deux camps [85] : les pro-codasyl menés par Charles Bachman (généralement les praticiens des SGBD) et les pro-relationnel menés par Edgar Codd (généralement les chercheurs et scientifiques de la communauté de BD). Les pro-Codasyl reprochaient au modèle relationnel son aspect formel difficilement compréhensible par les utilisateurs et programmeurs. Les pro-relationnel reprochaient au modèle Codasyl sa complexité et son manque de flexibilité pour la représentation de scénarios du monde réel, la difficulté d'optimiser les programmes et la faible indépendance des données. Durant ces années 70, la communauté de recherche travailla activement sur la proposition de prototypes de SGBD relationnels. Des études ont montré la supériorité des performances des BD relationnelles comparées à celles des BD orientées enregistrements [76]. Ces études ont servi de base pour le développement de SGBD relationnels menés par d'importants projets industriels comme le projet *SystemR* (initié par le groupe de recherche d'*IBM*), le projet *Ingres* (initié par l'université de Berkeley) et le projet *Gamma* [76]. Ces projets ont été les précurseurs de plusieurs champs de recherches du domaine de BD et de plusieurs contributions comme le développement des langages de requêtes, la proposition des *propriétés ACID* (acronyme pour Atomicité, Consistance, Isolation et Durabilité) pour les transactions de BD, l'optimisation des requêtes ou l'indépendance des données et les vues [76, 8]. Le débat entre les pro-Codasyl et les pro-relationnel prit fin en 1984 lorsqu'IBM, grand détenteur du marché des SGBD, annonça l'apparition de son prochain SGBD DB2 (successeur de l'IMS) développé selon le modèle relationnel. Le langage *SQL* (structured Query Language) a été adopté comme langage standard d'interrogation et de manipulation du modèle relationnel [85].

1.2.1.3 Méthodologie unifiée pour la conception BD

Malgré l'engouement suscité par le modèle relationnel, les éditeurs de logiciels et les utilisateurs en entreprises ont constaté que les modèles existants ne répondaient pas suffisamment à leurs attentes. Les éditeurs logiciels réclamaient urgemment des modèles de données incorporant davantage de sémantique et la possibilité d'intégrer des fichiers de formats variés dans leurs BD. Les utilisateurs en entreprise souhaitaient avoir des modèles de données pouvant exprimer toute la sémantique représentée par les règles de gestion de leur entreprise, et souhaitaient avant tout, avoir une méthodologie unifiée pour la conception de leurs BD [45]. C'est dans ce contexte que *Peter Chen* proposa en 1976 [46] son modèle *entités/associations* (E/A). Ce modèle a, comme le relationnel, un fondement mathématique car il est basé sur la théorie des ensembles et les relations mathématiques. Il a de plus l'avantage de présenter un modèle plus proche et plus compréhensible par l'utilisateur (figure 1.6). D'un point de vue formalisation mathématique, une principale différence est remarquée entre le modèle relationnel et le modèle E/A : le modèle

FIGURE 1.6 – Un exemple de diagramme entités/associations

relationnel utilise le constructeur de relation mathématique (produit cartésien) pour décrire la structure de *valeurs de données*. Une table est ainsi définie comme un produit cartésien des domaines de ses attributs. Le modèle E/A par contre, utilise le même constructeur pour décrire la structure *des entités*. Dans le modèle E/A, une relation est le produit cartésien des entités [45]. On remarque bien que le modèle E/A apporte un autre niveau d'abstraction car il traite les entités du mode réel et ne considère pas seulement les valeurs de données. Le modèle E/A apporte de plus, davantage de sémantique que le modèle relationnel comme par exemple la représentation des cardinalités dans le modèle E/A, et aussi le lien entre les entités qui est explicitement représenté dans le modèle E/A mais qui est implicite dans le modèle relationnel [45].

Malgré ces avantages, le modèle E/A n'a pas été un modèle de données implémenté dans un SGBD probablement selon [85] parce qu'il n'a pas été accompagné d'un langage de requêtes ou parce qu'il a été dépassé par le succès du relationnel. Le modèle E/A a cependant rapidement trouvé échos dans la communauté de conception des BD marquant une nouvelle évolution par l'introduction d'un niveau d'abstraction plus élevé que le niveau physique et logique, qui est le niveau conceptuel. *Peter chen* proposa dans ces articles une méthodologie pour construire un schéma initial E/A. Le modèle E/A est devenu très populaire dans les outils de conception de BD dans lesquels la traduction du schéma E/A en une collection de tables en troisième forme normale a été automatisée. Divers exemples d'outils ont été proposés comme *Silverrun*, *ERwin* et *ER/Studio*. Plusieurs grandes conférences ont été organisées autour du thème de la modélisation conceptuelle comme la *conférence ER* dont la première édition a été organisée en 1979. Ce thème de la modélisation conceptuelle a été porté par l'apparition d'autres modèles dits "sémantiques" qui se voulaient plus expressifs et comportant davantage de sémantique que le modèle relationnel. Les modèles les plus marquants de cette famille sont les modèles proposés par *Smith et al.* [182] et aussi *Hammer et McLeod* [82]. Les modèles objets ont suivi, basés sur les concepts du modèle E/A [45] pour apporter des aspects dynamiques aux modèles statiques existants.

1.2.1.4 Bilan 1 : évolution verticale des modèles de données

Comme premier bilan, nous remarquons que les modèles de données peuvent être représentés en trois grandes familles de modèles. Leur développement a marqué d'importantes évolutions dans le domaine des BD. La première famille de modèles comporte les modèles *orientés enregistrement* (hiérarchique et codasyl). Le modèle *relationnel* a apporté une indépendance physique des données ainsi qu'une puissance de calcul en introduisant des opérations pouvant s'effectuer sur des ensembles de données. Les modèles *conceptuels et sémantiques*, ensuite les modèles *objets* ont assuré une indépendance logique des données tout en apportant des modèles plus flexibles et plus expressifs. La flexibilité permet au modèle de faire face aux scénarios du monde réel. L'expressivité désigne la manière dont le modèle peut être en mesure de mettre en évidence les

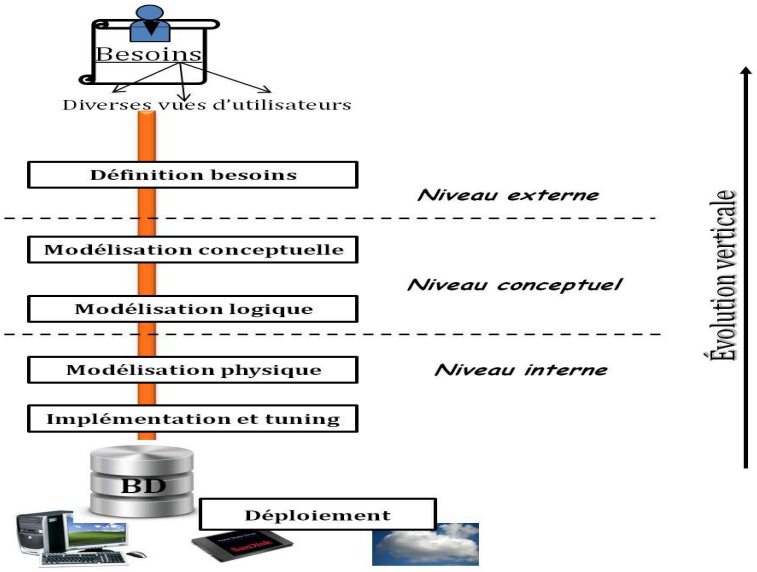

FIGURE 1.7 – Evolution verticale du cycle de conception des BD

différentes abstractions et les relations d'un scénario donné [142].

L'évolution de ces modèles de données que nous appelons évolution "verticale" (figure 1.7), a été unifiée dans l'architecture ANSI/SPARC [195] qui est actuellement universellement admise dans la communauté des BD. Cette architecture distingue les trois niveaux d'abstraction : le niveau conceptuel, externe et interne (figure 1.8) :

- Le niveau *conceptuel*, où la connaissance du domaine est formalisée dans un formalisme de modélisation conceptuel (E/A, modèle sémantique ou autre).
- Au-dessus, les modèles *externes* ou les vues utilisateurs qui permettent d'adapter les données fournies aux besoins des différentes catégories d'utilisateurs.
- En dessous, le modèle *interne* présente une spécification des données telle qu'elle sera implémentée dans le système de BD.

L'architecture ANSI/SPARC permet d'assurer l'indépendance logique et physique des données, i.e la capacité de modifier le modèle à un niveau de la BD sans avoir à modifier le modèle au niveau supérieur [55]. L'indépendance logique des données signifie la capacité de modifier le schéma conceptuel de la BD sans avoir à modifier ses schémas externes ou ses programmes d'applications. L'indépendance physique des données garantit la capacité de modifier le schéma interne de la BD (exp. Création de nouvelles structures d'accès ou réorganisation des fichiers de la BD) sans avoir à modifier ni son schéma conceptuel, ni les schémas externes.

1.2.2 L'évolution interne du cycle de conception

Se basant sur les niveaux de l'architecture ANSI/SPARC, un cycle de conception des BD a été défini, comprenant cinq principales étapes [142] : la définition des besoins, la phase de modélisation conceptuelle, la phase de modélisation logique, la phase de modélisation physique et la phase d'implémentation et de tuning. Ces étapes correspondent aux niveaux de l'architecture ANSI/SPARC de la manière suivante : le niveau externe et le niveau conceptuel traduisent les

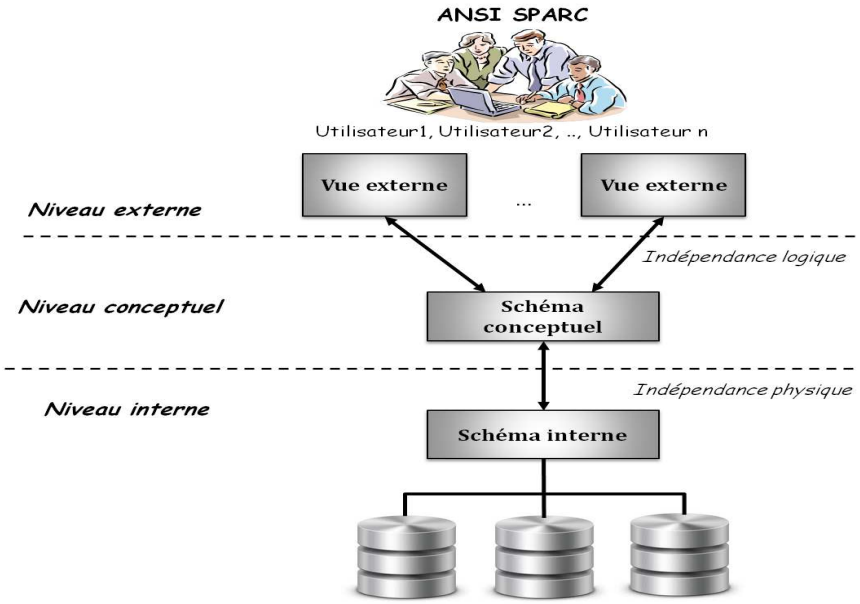

FIGURE 1.8 – L'architecture ANSI/SPARC

phases de modélisation conceptuelle et logique et le niveau interne correspond à la phase de modélisation physique [142, 55]. Les phases de ce cycle de conception ont été détaillées par la spécification des étapes de chaque phase (figure 1.9).

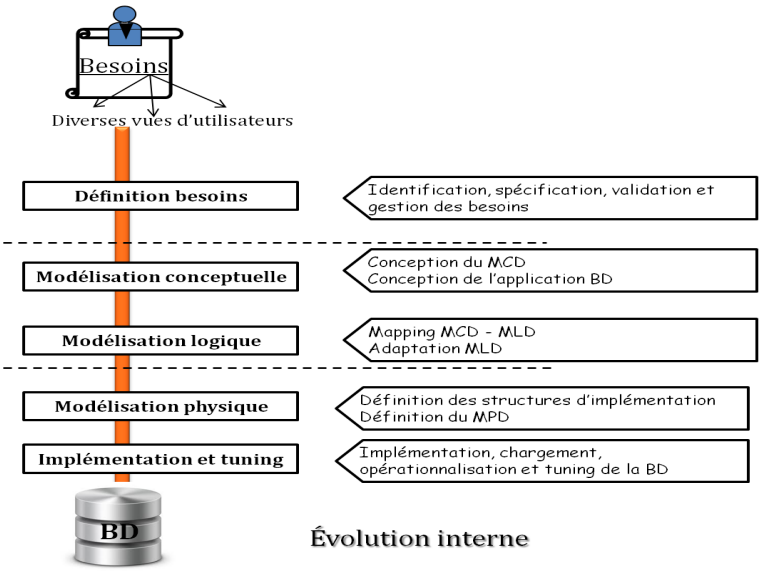

FIGURE 1.9 – Evolution interne du cycle de conception des BD

1.2.2.1 La définition des besoins

Le domaine relatif à la définition des besoins des utilisateurs est le domaine de l'*ingénierie des besoins* (IB). *Rolland* [159] définit l'lB comme suit : "l'ingénierie de besoins est concernée

par l'identification de buts assignés au système envisagé et l'opérationnalisation de ces buts en contraintes et exigences imposées au système et aux agents qui en assureront le fonctionnement. L'IB peut être vue comme le processus qui permet de transformer une idée floue en spécification précise des besoins servant de support à la spécification du système et de ses interfaces avec l'environnement".

Cette étape de "définition des besoins" consiste à définir de manière détaillée, l'ensemble des besoins et exigences des utilisateurs relatifs aux *données* et aux *traitements* [55]. En effet, parallèlement à la spécification des exigences relatives aux données, il est nécessaire de définir les exigences (fonctionnelles et non fonctionnelles) précisant l'ensemble des opérations et traitements que devra supporter l'application de BD. La phase de définition des besoins est assurée par un analyste ou un expert en ingénierie des besoins. Cette phase nécessite l'accomplissement de certaines tâches préliminaires comme : l'identification de la portée de l'application de BD ainsi que de l'ensemble des ses utilisateurs, l'étude de l'environnement du système incluant l'analyse des types de traitements et de leur fréquence, l'analyse des flux d'informations du système, des entrées et sorties des traitements, des caractéristiques géographiques des utilisateurs, l'étude et l'analyse de la documentation existante relative à l'application de la BD et l'établissement des questionnaires destinés aux utilisateurs. La phase de définition des besoins comprend les quatre étapes suivantes [183] :

- *La collecte des besoins* : les besoins sont collectés auprès des utilisateurs finaux ou des clients du système. Plusieurs techniques de collecte des besoins peuvent être utilisées comme les interviews, les réunions ou les workshops.
- *La spécification des besoins* : consiste à formaliser les besoins et à identifier leurs caractéristiques attendues. L'étape de collecte des besoins fournit généralement un premier ensemble de besoins informels, inconsistants ou incomplets. Trois principales techniques de spécification des besoins sont utilisées pour spécifier les besoins collectés d'une manière plus structurée [123] :
  1. *Les techniques informelles* : sont construites en langue naturelle avec ou sans règles de structuration. Leur usage introduit des risques d'ambiguïtés car ni leur syntaxe ni leur sémantique ne sont parfaitement définies. Parmi ces techniques, nous citons : le *questionnaire* et le *cahier des charges.*
  2. *Les techniques semi-formelles* : sont généralement basées sur des notations graphiques qui ont une syntaxe précise et permettent d'avoir une vision claire du système. Ces modèles sont de bons vecteurs de communication entre les concepteurs et les utilisateurs du système. Parmi ces techniques, nous citons les modèles de la méthode *Merise* et les modèles *UML* (comme les cas d'utilisations ou les diagrammes de séquence pour spécifier les besoins relatifs aux traitements).
  3. *Les techniques formelles* : sont basées sur des notations mathématiques qui fournissent un cadre précis et non ambigu pour la modélisation des besoins. Nous pouvons citer la spécification des *méthodes B* et *EB3* (Entity-Based Black-Box).
- *La validation des besoins* : consiste à valider le modèle obtenu lors de l'étape de spécification des besoins par les experts du domaine et les utilisateurs. Cette phase permet d'éviter la propagation des erreurs ou inconsistances des besoins durant les étapes de conception et d'implémentation de la base, qui peuvent s'avérer très coûteuses à corriger par la suite. Des

outils et langages peuvent être utilisés pour faciliter les tâches de validation des besoins. Des langages formels comme le langage *OCL* ou le langage *Z* sont utilisés pour compléter la spécification des besoins afin de vérifier leur consistance [55].

– *La gestion des besoins* : inclut toutes les activités permettant d'établir et de maintenir l'intégrité des besoins pendant que l'application de BD évolue. Quelques outils de traçabilité sont disponibles pour vérifier la traçabilité des besoins dans le cas où ces derniers évoluent très fréquemment.

1.2.2.2 La modélisation conceptuelle

Cette phase prend comme entrée la spécification des besoins collectés auprès des utilisateurs du système. Cette phase est assurée par les analystes et concepteurs du système. Elle comprend deux principales étapes menées en parallèle : la conception du modèle conceptuel et la conception de l'application et des transactions [55].

– *La conception du modèle conceptuel* : cette étape consiste à élaborer le modèle conceptuel des données (MCD) décrivant les exigences des utilisateurs relatives aux *données*. Ce modèle comprend une description détaillée de la structure de la BD, exprimée en utilisant les concepts fournis par un modèle de données d'un haut niveau d'abstraction décrit indépendamment de toute contrainte d'implémentation. Plusieurs exemples de modèles de haut niveau peuvent être utilisés comme le modèle *E/A*, le *digramme de classes* d'UML, le modèle *Express*, etc. L'établissement de ce modèle conceptuel repose généralement sur un dictionnaire regroupant le détail de l'ensemble des données manipulées par le système.
– *La conception de l'application de BD* : les opérations et les transactions définies à partir des exigences des utilisateurs relatives aux traitements, sont utilisées pour spécifier les requêtes des utilisateurs de haut niveau. Cette étape permet de vérifier si le schéma conceptuel défini répond à toutes les exigences identifiées [55]. Les opérations peuvent être de trois différents types [55] : les opérations de restitution des données, les opérations de mise à jour et les opérations combinant la restitution et les mises à jour. Ces opérations doivent être spécifiées dès le niveau conceptuel, en identifiant les entrées/sorties de chaque opération et leur comportement fonctionnel [55]. Cette étape de conception peut inclure une tâche d'identification des processus qui consistent en un ensemble d'opérations complexes. Des outils et notations sont utilisés pour spécifier les processus comme *BPwin*, les outils de modélisation de *workflow*, certains diagrammes *UML* (comme le diagramme d'activité et le diagramme de séquence), ou certains modèles de la méthode *Merise* qui permettent de représenter les traitements de l'application au niveau conceptuel comme le modèle conceptuel de traitements.

1.2.2.3 La modélisation logique

L'étape de modélisation logique prend en entrée le modèle conceptuel des données et fournit en sortie un modèle logique des données. Ce dernier représente l'organisation des données dans un modèle de données qui peut être implémenté. Cette organisation des données est également nommée déploiement logique des données. Le modèle logique de données possède des constructeurs de modélisation ignorants les détails physiques d'implémentation, et qui sont faciles à

comprendre par les utilisateurs. La traduction du modèle conceptuel vers un modèle logique se fait de manière directe voire automatique, en suivant des règles de traduction prédéfinies. Certains auteurs considèrent une phase additionnelle consistant en le "Choix du SGBD" comme phase qui précède la phase logique [55]. Ce choix du SGBD peut dépendre de facteurs techniques, économiques et stratégiques, il peut être une contrainte à respecter dès le début de la conception. L'application de BD peut aussi être conçue de manière générique pour pouvoir être déployée sur plusieurs plateformes de déploiement. Le choix du SGBD peut se faire dans ce cas lors de la phase de déploiement où l'administrateur choisit la plateforme qui répond le mieux aux besoins de l'application. La phase de modélisation logique se fait en deux principales étapes :

- *Mise en correspondance du modèle* : la première étape consiste à traduire le modèle conceptuel vers un modèle logique indépendamment des caractéristiques du SGBD.
- *Adaptation du modèle* : une deuxième étape consiste à adapter le schéma logique obtenu aux spécificités (constructeurs de modélisation et contraintes) du ou des SGBD sur lesquels sera implémenté le modèle logique. Le modèle logique est décrit à la fin de cette phase dans un *langage de définition des données* (LDD), il peut être complété lors de la phase de modélisation physique. Plusieurs outils de conception permettent de générer automatiquement le modèle logique décrit selon un LDD à partir d'un modèle conceptuel de données comme *ERwin*, *BPwin*, *Rational Rose* et *Visio*.

1.2.2.4 La modélisation physique

La phase de modélisation physique prend en entrée le modèle logique de données décrit selon un LDD et fournit en sortie un modèle physique des données (MPD) correspondant aux SGBD choisis pour l'implémentation de la BD. Ce modèle physique est défini dans un *langage de définition du stockage* (LDS). Cette phase nécessite une connaissance détaillée des paramètres de configuration, des structures physiques, des types de données ainsi que le langage de définition des données utilisé par le SGBD. Cette phase est généralement assurée par l'administrateur du système. Le but de la conception physique consiste à fournir une variété de choix pour le stockage des données en termes de regroupement, de partitionnement, d'indexation, etc [142]. Les choix des structures et chemins d'accès les plus pertinents se font selon plusieurs critères comme le temps de réponse aux requêtes, l'espace de stockage utilisé par les fichiers de la base ou le débit moyen des transactions. Les requêtes et transactions sont dans ce cas celles identifiées et spécifiées lors de la phase de définition des besoins et de modélisation conceptuelle.

Afin de respecter les trois niveaux imposés par l'architecture ANSI/SPARC, deux principaux langages de données sont utilisés pour assurer l'indépendance logique et physique des données : le LDD est utilisé pour spécifier le schéma conceptuel et logique, le LDS est utilisé pour spécifier le schéma interne. Dans la plupart des SGBD actuels, il n'existe pas de langage spécifique qui joue le rôle de LDS. Le schéma interne est alors défini par une combinaison de fonctions, de paramètres et de spécifications relatives au stockage des données. Un troisième langage devrait également exister pour définir les vues des utilisateurs ; mais la plupart des SGBD utilisent le LDD pour définir les schémas conceptuels et externes. Dans les SGBD actuels, les trois types de langages cités ne sont généralement pas considérés comme des langages distincts, mais un langage global et intégré est utilisé qui comporte des constructeurs pour la définition des différents schémas. Pour l'exemple des BD relationnelles, le langage SQL est utilisé en tant que langage de définition

des données, de définition des vues et de manipulation des données. Le LDS est un langage qui existait dans les premières versions de SQL, mais a été retiré du langage SQL afin de le maintenir aux deux niveaux conceptuel et externe uniquement [55].

1.2.2.5 La phase d'implémentation et de tuning

La dernière phase d'implémentation comprend deux principales étapes [55] :

- *Implémentation* : l'étape d'implémentation consiste à implémenter le schéma physique de données et à procéder au chargement des données de la BD. Les programmes d'application de la base sont générés à partir de la spécification conceptuelle des opérations et transactions (identifiée lors de la phase de modélisation conceptuelle). Les programmes d'application sont implémentés en utilisant un langage de manipulation des données (LMD). Le LMD peut être déclaratif ou procédural, il est utilisé pour spécifier la recherche, l'insertion, la suppression et la modification des données.
- *Opérationnalisation de la BD* : la deuxième étape consiste à rendre la BD opérationnelle pour les utilisateurs et à assurer sa maintenance dans le cas où les besoins évoluent. Une étape de *tuning* peut être envisagée pour la maintenance de la base. Le tuning peut se faire tout au long du cycle de vie de la BD, au fur et à mesure que la base et l'application évoluent. Afin de réduire l'ensemble des tâches des administrateurs qui sont coûteuses pour une entreprise, une étape de *self-tuning* peut être considérée dans ce cycle de conception.

Une autre phase annexe peut aussi être ajoutée à ce cycle de conception, qui est la phase de *personnalisation*. La richesse des BD et la masse de données qu'elles stockent ont influencé leur exploitation par les utilisateurs du fait que ces derniers ont des préférences et des profils souvent liés au contenu de la BD. En conséquence, la phase de personnalisation est ajoutée pour gérer ces profils. La figure 1.10 décrit les étapes des phases de conception dans un schéma E/A selon la notation de Peter Chen.

1.2.3 Evolution horizontale des modèles de données

Le cycle de conception des BD établi a été marqué par un autre type d'évolution, que nous appelons évolution "horizontale". Cette évolution est représentée par la diversification des modèles conceptuels, logiques et physiques qui sont apparus pour répondre à des types d'applications spécifiques (figure 1.11). Nous retraçons dans ce qui suit les modèles les plus marquants.

1.2.3.1 Les SGBD orientées objet

L'apparition de plusieurs modèles sémantiques a marqué une évolution importante dans le cycle de conception des BD, par l'introduction de la phase de modélisation conceptuelle comme phase à part entière. Cependant, ces modèles sémantiques n'ont pas été implémentés au sein de la BD, ce qui a fait émerger des propositions permettant d'exporter la sémantique apportée par ces modèles au niveau logique. Les SGBD orientés objet (OO) sont un exemple type de ces propositions. Ces SGBD ont connu un grand intérêt dans le milieu des années 80. Leur premier objectif était alors de gérer le problème de dysfonctionnement ou "impedance mismatch" entre les langages de programmation orientés objet comme le langage C++ et les bases de données relationnelles [85], qui consiste à encapsuler des structures dont les attributs ont des types qui ne

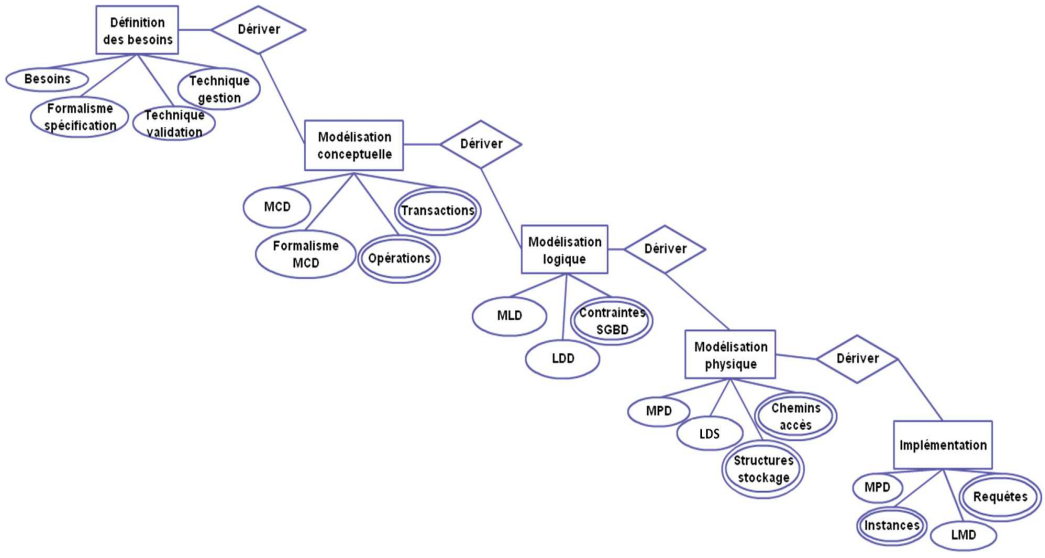

FIGURE 1.10 – Evolution interne : les étapes des phases du cycle de conception

sont pas les mêmes que ceux du langage de programmation. La résolution de ce problème implique des conversions de types ainsi que des contrôles de types qui compliquent la programmation. Pour lier une application développée en C++ (par exemple) à une BD relationnelle, il fallait adapter le langage de programmation au langage du modèle de données, ou le contraire. Comme les langages de programmation étaient suffisamment évolués, les SGBD objet ont été proposés et basés sur un modèle de données supportant les concepts des langages de programmation orientés objet. Le but était de fournir un stockage persistant pour les objets et les structures de données utilisés par les développeurs dans leurs programmes, qui étaient habituellement perdus une fois que le programme se termine [55]. Comme dans tous les langages de programmation, il est possible de sauvegarder les objets dans des fichiers pour prolonger leur durée de vie. Cependant, ce type de sauvegarde fournit une première forme de persistance des objets qui assez primitive. Nous retrouvons ainsi tous les problèmes engendrés par la gestion des données sur des fichiers classiques. L'approche base de données OO apporte une solution à la gestion transparente de la persistance des objets.

Malgré l'intérêt suscité par les SGBD orientés objet, ces derniers n'ont pas réussi à percer dans le marché des BD où leur taux d'utilisation dans les applications de BD avoisine les 5% [55]. Cet échec peut s'expliquer par diverses raisons : les différentes offres de fournisseurs de SGBD OO n'étaient pas compatibles, aussi ces SGBD offraient peu de support aux transactions et aux requêtes contrairement aux SGBD relationnels [85]. Cependant, le besoin d'ajouter de nouvelles structures et types de données aux SGBD relationnels a permis l'évolution des SGBD existants par l'intégration de nouvelles fonctionnalités. Certaines fonctionnalités ont été créées, sous la forme de modules optionnels, spécifiquement destinés à un type d'application comme les applications de traitements d'images ou les applications spatiales [55]. D'autres fonctionnalités avaient un objectif plus général, telles que l'incorporation de concepts orientés objet dans les systèmes relationnels. Ceci a donné naissance aux SGBD relationnels objets. Cette approche a été initiée

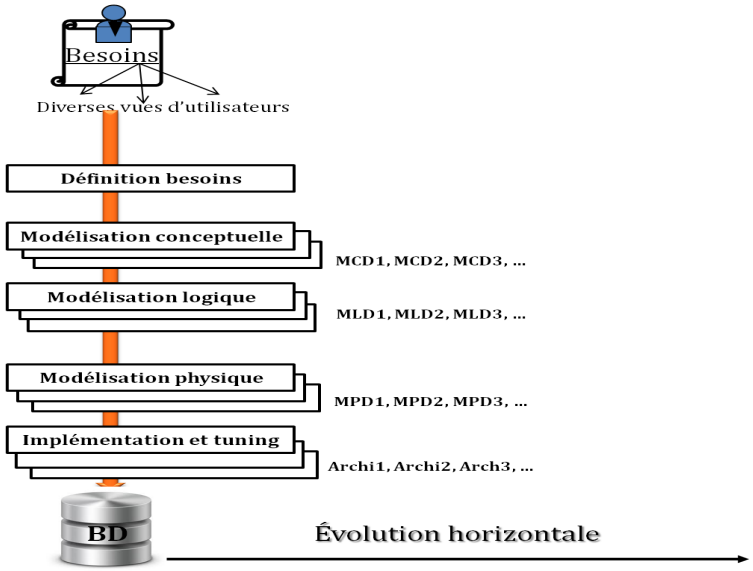

FIGURE 1.11 – Evolution horizontale du cycle de conception des BD

par le projet *Ingres* (évoluant en Postgres pour post-Ingres) en 1972 à l'université de Berkeley [76]. Ce projet portait sur les systèmes d'informations géographiques et un de ses objectifs était alors de modéliser les données géographiques dans une BD relationnelle. Une difficulté importante a été remarquée lors de l'expression de requêtes comportant des concepts géographiques (emplacement, dimension, etc) sous forme de requêtes SQL. Les systèmes relationnels ont été conçus pour supporter des types de données classiques et non les types géographiques. Cette observation est valable pour d'autres domaines nécessitant des types de données spécifiques. Ces constats ont motivé la proposition de premiers SGBD relationnels objets (comme *PostgreSQL*), qui ont permis la définition de types de données, d'opérateurs, de fonctions et de méthodes définis par l'utilisateur [55, 172]. Nous avons assisté par la suite et même actuellement au développement d'une panoplie de différents types de SGBD spécifiques à des types d'applications (comme les SGBD orientés *graphe*, ou plus récemment les SGBD *NoSql*). Ces propositions poursuivent le même objectif général que les SGBD objet, qui est d'offrir des modèles de données plus flexibles et plus adaptés à des besoins spécifiques des utilisateurs, programmeurs et concepteurs.

1.2.3.2 Diversification des modèles conceptuels

Avec la popularisation des applications web, de nouveaux modèles conceptuels ont été proposés au début des années 2000 pour représenter les données semi-structurées. Ces modèles sont principalement représentés par la famille des modèles XML (eXtended Markup Language). Diverses techniques ont été développées dans le cadre des applications web pour permettre l'échange de données sur le web. Actuellement, XML est considéré comme la principale norme permettant l'échange de données entre les différents types de BD et les pages web. XML combine les concepts des modèles utilisés dans les systèmes de documents avec des concepts de modélisation de BD [55]. Dans un modèle classique, le schéma de données est d'abord défini. Les instances sont créées conformément à ce schéma. Dans un modèle XML (exp. XMLSchema), le schéma n'a pas besoin

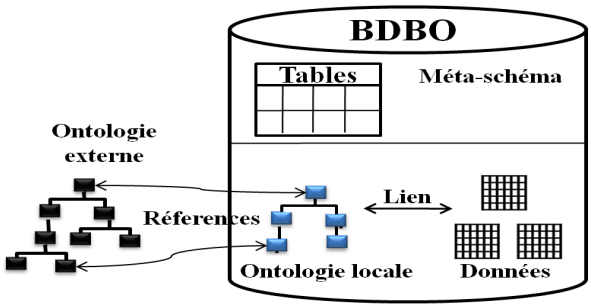

FIGURE 1.12 – Architecture d'une BDBO

d'être spécifié à l'avance [85]. Les instances se décrivent par elles-mêmes, où chaque attribut est marqué par une métadonnée décrivant le sens de cet attribut. Cette famille de modèles marque un retour vers les modèles conceptuels. *Peter Chen*, développeur du modèle E/A, a d'ailleurs exercé comme expert pour le groupe XML [85]. Là encore, les SGBD XML (basés sur les SGBD orientés documents) sont apparus pour exporter la sémantique de ces modèles au niveau logique et physique (au sein de la BD). Les modèles *ontologiques* du web sémantique ont suivi, définis pour la plupart sur la base du modèle XML (exp. *RDF*, *RDFS*, *OIL*, *DAML*, *OWL*). *Tim Berners Lee*, développeur du modèle *RDF*, considère son modèle comme faisant partie de la famille des modèles E/A ou plus généralement des modèles conceptuels. Nous rappelons qu'une *ontologie* est définie comme "une spécification formelle et explicite d'une conceptualisation partagée d'un domaine de connaissance" [78]. Ces dernières années, les ontologies ont été utilisées dans plusieurs applications telles que l'annotation de documents, la gestion de catalogues de composants industriels, l'intégration de données, etc. Cette intense utilisation des ontologies a donné lieu à un nouveau type de données appelées *données sémantiques* ou *données ontologiques*, qui sont définies comme des données référençant des ontologies. Quelques systèmes ont proposé de gérer ce type de données en mémoire centrale comme *OWLIM* [111] ou *Jena* [137]. Ces systèmes facilitent le chargement et la mise à jour des données mais deviennent impraticables lorsqu'il faut gérer un grand volume de données ontologiques. Pour assurer le passage à l'échelle, de nombreux systèmes ont proposé de gérer les données ontologiques au sein de BD qui proposent des mécanismes efficaces pour le stockage et l'interrogation des données. Cette nouvelle architecture de base de données est appelée *Base de Données à Base Ontologique* (BDBO). Nous avons constaté que la diversification des modèles de données au niveau logique et physique avait comme premier objectif d'exporter davantage de sémantique au sein de la BD, et davantage de flexibilité pour les utilisateurs. Nous remarquons que le développement des BDBO s'inscrit dans la même lignée, où toute la sémantique des données de la base est représentée et persistée au sein de la BD sous forme d'une ontologie locale (figure 1.12). Cette ontologie locale peut être considérée comme le modèle conceptuel de la BD, ou peut être utilisée pour extraire ce modèle conceptuel. Une nouvelle étape est franchie : ce n'est plus uniquement le modèle logique de données qui est représenté au sein de la base, c'est aussi son modèle conceptuel.

L'émergence des BDBO a fait apparaître de nouveaux modèles de données physique et de nouvelles architectures de BD. Contrairement aux BD classiques, où le modèle logique est stocké selon une approche relationnelle, dans une BDBO différents modèles de stockage physique (représentation horizontale, spécifique, etc.) sont utilisés pour stocker les deux niveaux de modé-

lisation : le niveau ontologie et le niveau des instances ontologiques. L'architecture des systèmes gérant les bases de données a aussi évolué : architecture monocouche où les données sont manipulées au niveau de la mémoire centrale (main memory databases), architecture à deux couches représentant la méta-base et le contenude la base (utilisée par la majorité des systèmes), architecture à trois couches et architecture à 4 couches. Nos contributions dans cette étude reposent sur l'utilisation des BDBO. Nous détaillerons donc les notions relatives aux BDBO dans la deuxième partie de ce manuscrit.

1.2.3.3 La technologie évolue aussi

Parallèlement au développement des différents modèles de données proposés, les recherches sur les aspects matériels ont mené au développement de nouvelles architectures de déploiement. Pendant les années 90, de nouvelles recherches ont étudié les BD réparties géographiquement et les possibilités d'accès en parallèle aux données. Les travaux théoriques sur les BD distribuées ont conduit à la définition de plusieurs prototypes. Le projet *Gamma* par exemple, a été initié afin d'explorer la capacité des disques à traiter les données en parallèle et a mené au développement d'une machine de BD parallèle appelé *Gamma*. Les systèmes parallèles d'*IBM*, *Tandem*, *Oracle*, *Informix*, *Sybase*, et *AT&T* sont basées sur les recherches du projet *Gamma* [76]. Aujourd'hui, la majorité des systèmes de BD offrent la possibilité de distribuer et de répliquer les données entre les nœuds d'un réseau informatique [76]. Par conséquent, avec l'évolution de la technologie des BD et le développement de réseaux et les systèmes de stockage et de traitement, la phase d'implémentation du cycle de conception des BD doit être considérée avec importance, et doit prendre en compte diverses architectures matérielles sur lesquelles les BD sont déployées (bases de données distribuées, parallèles, cluster de bases de données, grille de calcul, bases de données sur flash, cloud, etc.).

1.2.3.4 Bilan 2 : évolution horizontale du cycle de vie

Nous avons assisté à une troisième vague d'évolution "horizontale" des modèles de données où chaque phase du cycle de conception des BD a été étendue par de nouveaux modèles. Au niveau conceptuel, les modèles objet, les modèles XML et les modèles ontologiques ont montré leur contribution. La non implémentation des modèles conceptuels dans les SGBD a permis de proposer de nouveaux modèles de données logiques (objet, relationnel objet, spatial, etc) incorporant certaines sémantiques apportées par ces modèles conceptuels, pour être représentées au sein de la BD. Dans la même continuité, le développement des BDBO a permis la définition de nouveaux modèles de données pour stocker au niveau physique les instances sémantiques et les ontologies décrivant leurs sens. L'évolution technologique a par ailleurs permis le développement de nouvelles architectures matérielles d'implémentation et de déploiement des BD. La figure 1.13 illustre sous forme d'un diagramme d'activité, le flux des étapes de chaque phase du cycle de conception, les entrées/sorties de chaque phase ainsi que les acteurs intervenant au sein de chaque phase.

Même si le cycle de conception a connu divers types d'évolutions, la définition détaillée et la stabilité des phases du cycle de conception ont constitué une feuille de route à suivre pour le développement de plusieurs types de BD par diverses entreprises et organisations. Un volume

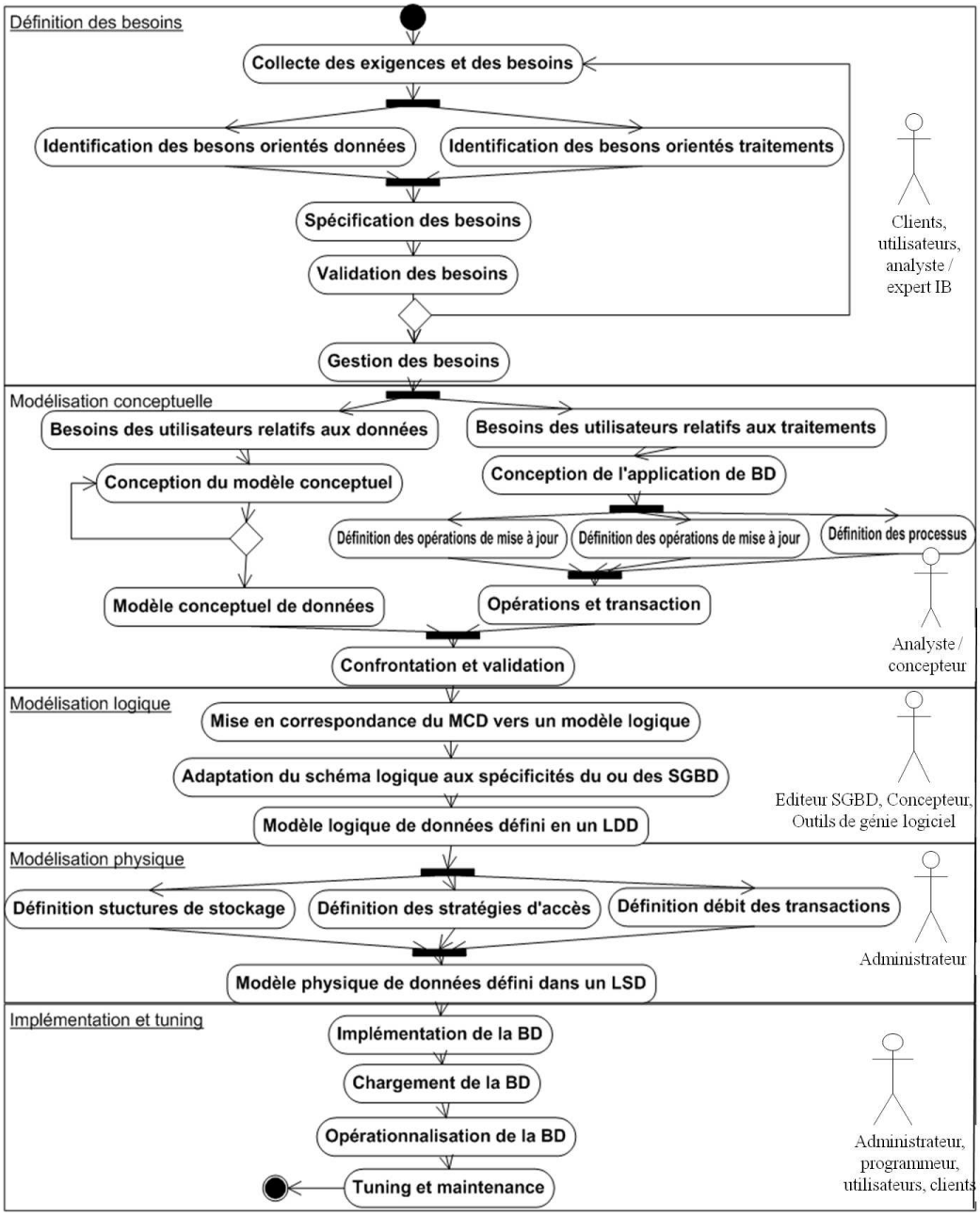

FIGURE 1.13 – Cycle de conception des BD : flux d'activités, étapes et acteurs

important de données a ainsi été produit et géré par les BD développées. Un nouveau besoin a été ressenti par ces entreprises consistant en l'analyse de cette quantité de données. Les *entrepôts de données* (ED) ont été la solution proposée pour répondre à ce besoin. Nous détaillons dans ce qui suit les notions relatives aux ED, et nous montrons que leur cycle de conception s'inscrit dans une démarche de consolidation du cycle de conception défini pour les BD.

1.3 Des BD aux ED : cycle de conception des entrepôts de données

1.3.1 Les ED : Explosion des données et besoin d'analyse

Pendant les années 80, les entreprises et les organisations ont eu à gérer des mégaoctets voire des giga-octets de données. Dans les années 90, ce volume de données a augmenté pour atteindre les téraoctets et pétaoctet de données [55]. Ces données étaient stockées dans des BD disparates présentant des *ilots d'informations.* L'augmentation continuelle de ce volume de données a créé un nouveau besoin qui est celui d'*analyser* ces quantités de données afin d'en extraire de nouvelles connaissances. Par exemple, la technique connue des ventes croisées présentant une technique de marketing permettant de proposer des produits ou des offres complémentaires à un achat ou une consultation d'un produit. L'extraction de ce type de connaissances nécessite d'avoir une vue unifiée et intégrée des données issues des BD hétérogènes qui se trouvent au sein de la même organisation ou bien dans différentes organisations. [55]. Un *système d'intégration* (virtuel ou matérialisé) est la solution permettant de garantir cette intégration. Pour assurer une séparation entre les données spéculative et les données représentant l'état actuel de l'organisation, la solution choisie a été de dupliquer les données dans une nouvelle base de type "entrepôt de données" (ED). Le concept d'ED a rapidement trouvé échos dans divers domaines et applications, et les premiers ED ont été développés au sein de projets industriels. La technologie des ED est actuellement déployée avec succès dans diverses industries : production, vente au détail, services financiers, transports, télécommunications, médecine, etc. Des outils d'analyse sont utilisés en complément aux ED afin de fournir diverses techniques d'analyse, de visualisation et de consolidation des données de l'ED. Au début des années 90, le concept de *traitement d'analyse en ligne* (OLAP) a été proposé par *Edgar Codd* en 1993 [49] pour décrire les fonctionnalités de l'outil *Essbase* de la société Arbor Software rachetée plus tard par Oracle, outil qui offre différentes fonctionnalités d'analyse pour les BD multidimensionnelles. Plusieurs autres outils OLAP ont suivis. Un ensemble de règles ont été spécifiées définissant les règles que doivent assurer les ED pour supporter une analyse OLAP [47]. La plupart des nouveaux produits orientés "entrepôt de données" ont par la suite été appelés outils de l'informatique décisionnelle (business intelligence). Nous définissons dans ce qui suit les principales notions relatives aux ED, et nous décrivons leur cycle de conception.

1.3.1.1 Définition et architecture

Plusieurs définitions ont été données pour le concept d'ED. Nous retenons la définition de *W.H. Inmon*, considéré comme le père des ED qui décrit un ED dans son ouvrage de référence "Building the Data Warehouse" [89], comme "une collection de données orientées sujet, intégrées,

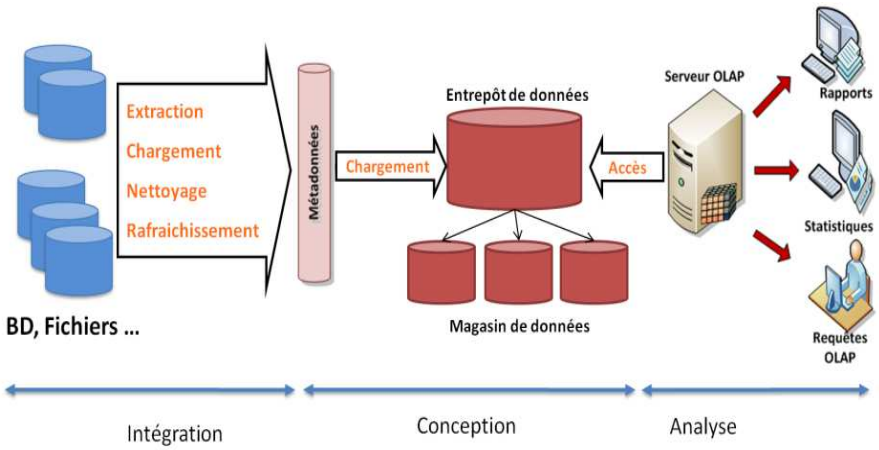

FIGURE 1.14 – Architecture d'un entrepôt de données

non volatiles et historisées, organisées pour supporter un processus d'aide à la décision". Cette définition englobe les termes clés suivants [89] :

- *Orientées sujet* : les données des systèmes d'informations classiques sont organisées selon les applications d'une entreprise. Ces données sont orientées applications. Les données d'un ED sont organisées par thèmes ou par sujets (par exemple : production, vente, marketing, etc).
- *Intégrées* : les données d'un ED proviennent de différentes sources hétérogènes donnant lieu à de nombreux conflits syntaxiques et sémantiques entre les données des sources. L'intégration des données permet d'éliminer l'ensemble de ces conflits afin d'avoir une représentation uniforme et cohérente des données lors de leur chargement au niveau de l'ED. Selon *W.H. Inmon* [89], de toutes les caractéristiques d'un ED, l'intégration est l'un des aspects les plus importants.
- *Non volatiles* : Les données d'un ED sont généralement utilisées en mode consultation. Elles peuvent être interrogées mais ne sont ni modifiées, ni supprimées (sauf dans les cas de rafraichissement de l'ED). Ceci permet de conserver la traçabilité des informations afin de pouvoir effectuer des analyses sur une longue période.
- *Historisées* : les données d'un ED doivent être associées à un référentiel temps afin de refléter l'activité de l'entreprise sur une période donnée.
- *Organisées pour supporter un processus d'aide à la décision* : les données provenant des sources doivent être agrégées afin de faciliter leur analyse. Ces données peuvent être consultées à travers des outils (requêtes, outils OLAP, outils de fouille de données, outils de statistiques, etc) permettant leur manipulation et leur analyse.

L'architecture d'un ED, comme illustrée dans la figure 1.14, peut être structurée en quatre axes [109] :

- *Les sources de données* : l'ED stocke des données provenant de différentes sources d'informations hétérogènes et distribuées. Ces sources peuvent être des BD, des fichiers de données, des sources externes à l'entreprise, etc.
- *Stockage* : avant de pouvoir être stockées, les données des sources doivent d'abord être nettoyées. Le processus de nettoyage consiste à sélectionner et à épurer les données pour

éliminer toute erreur et réconcilier les différences sémantiques entre ces données. Une fois nettoyées, ces données seront intégrées dans l'entrepôt. Le processus de rafraîchissement consiste à propager vers l'entrepôt, les changements effectués sur les données des sources. Les données concernant la création, la gestion et l'usage de l'entrepôt sont stockées dans un répertoire indépendant. Ces données sont appelées *métadonnées*. Les métadonnées peuvent contenir des informations sur les sources et leurs contenus, le schéma de l'ED, les règles de rafraîchissement, les profils et groupes d'usagers, etc. Un ED peut comporter plusieurs magasins de données. Ces derniers sont des extraits de l'ED consacrés à un type d'utilisateurs et répondant à un besoin spécifique. Ils sont dédiés aux analyses décisionnelles de type OLAP [108].

- *Serveur OLAP* : un serveur OLAP permet d'accéder à l'entrepôt, il convertit les requêtes des clients en requêtes d'accès à l'ED et fournit des vues multidimensionnelles des données à des outils d'aide à la décision.
- *Outils de front end* : ces outils formatent les données, conformément aux besoins des utilisateurs, sous différentes formes : requêtes OLAP, tableaux, courbes, rapports, statistiques, etc.

1.3.1.2 La modélisation multidimensionnelle

Les applications d'aide à la décision à base d'ED utilisent des processus d'analyse en ligne de données OLAP répondant à des besoins d'analyse de l'information. Pour ce type d'environnement OLAP, une nouvelle approche de modélisation a été proposée : la *modélisation multidimensionnelle*. Popularisée par *Ralph Kimball* dans les années 90, cette modélisation est aujourd'hui reconnue comme la modélisation la plus appropriée aux besoins d'analyse et de prise de décision [6]. La modélisation multidimensionnelle est une technique de conceptualisation et de visualisation des modèles de données. Elle offre une structuration et une organisation des données facilitant leur analyse. Elle a pour principal objectif d'avoir une vision multidimensionnelle des données. Les données sont organisées de manière à mettre en évidence le sujet analysé et les différentes perspectives d'analyse. Cette modélisation est intéressante d'une part parce qu'elle représente une modélisation assez proche de la manière de penser des analystes. D'autre part, elle facilite aux utilisateurs la compréhension des données. Un modèle multidimensionnel renferme les deux concepts fondamentaux de *fait* et de *dimension*.

- Un *fait* représente le sujet ou le thème analysé. Il présente un centre d'intérêt de l'entreprise et est considéré comme un concept clé sur lequel repose le processus de prise de décision. Un fait est formé de *mesures* ou attributs du fait (atomiques ou dérivés) qui correspondent aux informations liées au thème analysé. Par exemple pour la gestion des commandes, les mesures peuvent être la quantité du produit commandé et le montant de la commande.
- Une *dimension* représente un contexte d'analyse d'un fait. Par exemple la gestion des commandes peut être analysée selon les dimensions Client, Magasin, et Temps. Les dimensions se présentent sous forme d'une liste d'éléments organisés de façon *hiérarchique*. Par exemple, pour la dimension temps, nous pouvons avoir la hiérarchie suivante : année, semestre, trimestre, mois, semaine et jour. Les dimensions sont caractérisées par des attributs de dimensions. Une hiérarchie est dite complète si tous les objets d'un niveau de la hiérarchie appartiennent à une seule classe d'objets d'un niveau supérieur et forment cette

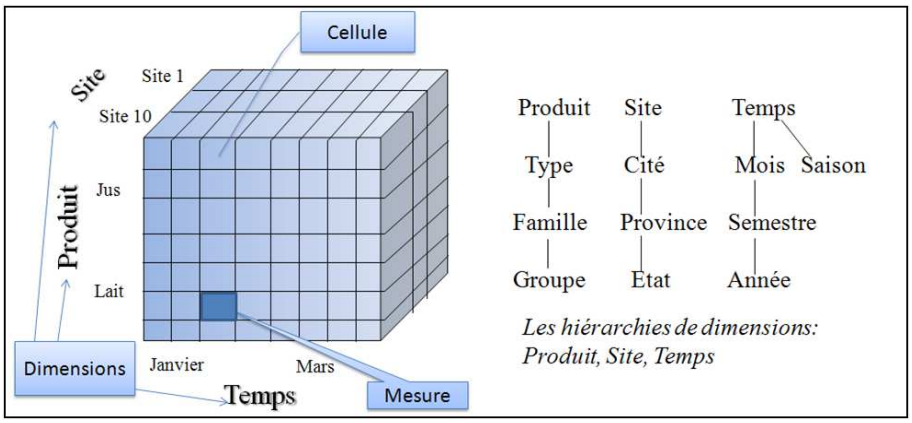

FIGURE 1.15 – Le cube multidimensionnel Vente

classe.

Le constructeur des modèles multidimensionnels est appelé un *cube* de données (figure 1.15). Un cube de données est composé de cellules qui représentent les mesures (les attributs du fait). Le cube ci-dessous permet d'analyser les mesures selon les différentes dimensions : produit, site et temps. Les hiérarchies définies sur une dimension peuvent être simples ou multiples.

Plusieurs opérations OLAP peuvent être effectuées sur cette structure multidimensionnelle appelées opérations de restructuration qui sont : *pivot* (rotation du cube autour d'un des axes), *switch* (interchanger la position des membres d'une dimension), *split* (couper le cube en réduisant le nombre de dimensions), *nest* (imbriquer des membres de dimensions), et *push* (combiner les membres d'une dimension aux mesures du cube). D'autres opérations sont liées à la granularité permettant ainsi la hiérarchisation des données. Ces opérations sont : *roll up* qui permet de visualiser les données de manière résumée en allant d'un niveau particulier de la hiérarchie vers un niveau plus général et l'opération *drill down* qui permet de naviguer vers des données d'un niveau inférieur et donc plus détaillé.

Plusieurs auteurs ont proposé des modèles de données adaptés au paradigme multidimensionnel pour le développement d'un ED, on peut citer : *Sapia et al.* [168] qui présentent le modèle ME/R (Multidimensional Entity Relationship) accompagné d'une représentation graphique qui est une spécialisation du modèle E/A, et l'adaptent pour la modélisation conceptuelle des systèmes OLAP. *Cavero et al.* [42] proposent un modèle IDEA, qui étend le modèle E/A par les concepts d'héritage, et de généralisation de hiérarchies afin de représenter les sémantiques multidimensionnelles. *Bulos et al.* [31] présentent une modélisation ADAPT se basant sur deux principaux objets multidimensionnels : les hyper cubes et les dimensions ainsi que d'autres objets supplémentaires. *Lujn-mora et al.* [126] proposent un modèle multidimensionnel se basant sur la notation UML. Différents packages on été introduits (package fait, package dimension et package étoile), représentés par des stéréotypes particuliers qui étendent la notation UML. *Abello et al.* [4] effectuent une classification des modèles de données multidimensionnels proposés dans la littérature. Cette classification couvre les différentes phases de modélisation proposées par les auteurs et les différents concepts utilisés.

1.3.1.3 Le cycle de vie des entrepôts de données

Le *cycle de vie* reconnu actuellement pour les ED inclut les phases suivantes [68, 108] :

- *Planification* : cette phase vise à préparer le terrain pour le développement de l'ED. Elle inclut les tâches suivantes :
  - déterminer l'étendue du projet ainsi que les buts et objectifs de l'entrepôt à développer,
  - évaluer la faisabilité technique et économique de l'entrepôt,
  - identifier les futurs utilisateurs de l'entrepôt.
- *Conception et implémentation* : cette phase consiste à développer le schéma de l'entrepôt et à mettre en place toutes les ressources nécessaires à son implémentation et à son déploiement.
- *Maintenance et évolution* : cette phase implique l'optimisation de ses performances périodiquement. L'évolution de l'ED concerne la mise à jour de son schéma en fonction des différents changements survenant au niveau des sources ou des besoins des utilisateurs.

La deuxième phase de *conception* comporte actuellement cinq principales phases, constituant le *cycle de conception* de l'ED : l'analyse de besoins, la modélisation conceptuelle, la modélisation logique, la processus d'extraction-transformation-chargement (ETL) et une phase de modélisation physique [68]. Ce cycle de conception doit suivre une modélisation multidimensionnelle tout au long des phases de conception citées. Nous remarquons que ce cycle de conception est très similaire au cycle de conception des BD. Cependant, avant d'atteindre le consensus concernant les phases du cycle de conception des ED, plusieurs recherches ont été menées pour converger vers ce cycle, que nous allons présenter.

La première génération des projets d'ED n'ont pas toujours suivi ce cycle de conception, principalement pour les deux premières étapes (*l'analyse des besoins* et *la modélisation conceptuelle*) qui ont été partiellement négligées [68, 69]. Plusieurs études ont montré par la suite la nécessité d'inclure une étape de modélisation conceptuelle offrant une vue abstraite du domaine étudié, et facilitant la communication entre les concepteurs et les utilisateurs de l'ED [68, 62, 158]. Il a également été reconnu par les praticiens du domaine et par la communauté de recherche, que la prise en compte des besoins présente un facteur-clé déterminant le succès ou l'échec du projet d'entreposage [158].

La *première génération* des projets d'ED, suivant la vision de *W.H. Inmon*, s'est concentrée sur les trois phases de modélisation logique, le processus ETL et la modélisation physique. Les aspects d'optimisation ont été largement étudiés dans ces travaux. Ceci s'explique par le fait que les projets d'entreposage sont issus originellement de l'industrie qui s'intéresse aux aspects pratiques et ne donne pas suffisamment d'importance aux problèmes de conceptualisation, et également par le fait que les modèles logique et physique d'un ED jouent un rôle important dans l'amélioration et l'optimisation des performances du système décisionnel final. La première étape de *modélisation logique* consiste à fournir un schéma logique de l'ED selon une plate-forme cible, intégrant l'ensemble des schémas des sources de données. Le *processus ETL* consiste à extraire les données des sources afin de peupler le schéma cible de l'ED. L'étape de modélisation physique consiste à implémenter le schéma de l'ED selon un SGBD cible. Le but d'un ED dans ces premiers projets est de fournir une plateforme uniforme aux données provenant de sources réparties et hétérogènes. L'aspect d'intégration est prépondérant. Cette vision fait d'un entrepôt, un cas particulier d'un système d'intégration matérialisant les données des sources.

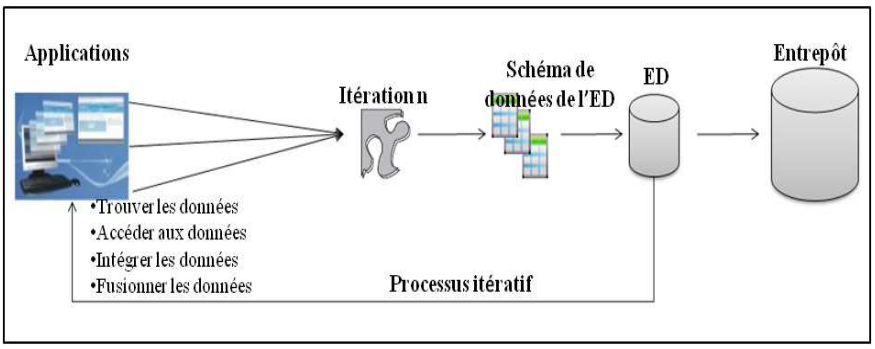

FIGURE 1.16 – Conception d'un ED par un processus d'intégration (selon Inmon)

Dans la *deuxième génération* des projets d'entreposage, la reconnaissance des deux phases d'analyse des besoins et de modélisation conceptuelle comme des phases indispensables, permet de compléter le cycle de conception. Nous exposons dans ce qui suit ses deux visions de conception d'un ED : la section suivante positionne l'ED comme un système d'intégration matérialisant les données des sources. Nous étudierons les problématiques liées à l'hétérogénéité des données et leur intégration. La section d'après récapitule les évolutions des projets d'ED qui ont mené à l'établissement du cycle de conception complet, incluant les deux étapes d'analyse des besoins et de modélisation conceptuelle.

1.3.2 Entrepôt de données vu comme un système d'intégration matérialisé

La problématique d'intégration des données est considérée comme un processus important dans les projets d'entreposage. La première génération des projets d'entreposage a considéré l'ED comme un système d'intégration matérialisé et s'est davantage intéressée aux problématiques liées à l'intégration des données [89]. D'ailleurs, *W.H. Inmon* décrit l'ED comme une collection de données intégrées et organisées pour supporter un processus d'aide à la décision [89]. Il y décrit le processus d'intégration comme une tâche importante et fournit une méthodologie de conception de l'ED principalement axée sur l'analyse et l'intégration des données des sources (sans forcément nommer cette étape "processus ETL"). Selon la vision d'*Inmon* [89], le schéma de l'ED est construit à partir des sources de données selon le processus itératif suivant (figure 1.16) : l'analyse des sources qui comprend diverses tâches consistant à trouver les données, accéder aux données, intégrer les données et fusionner ces données. Un modèle de données est généré à chaque itération, qui servira comme schéma de base pour l'itération suivante. A chaque itération, des tables de l'entrepôt sont conçues et permettent de construire une version de l'ED qui peut être validée par les utilisateurs. L'itération qui suit alimente l'entrepôt de l'itération précédente par de nouvelles données et organise davantage ses données. D'autres problématiques liées à l'optimisation des performances de l'ED ont également été étudiées dès les premiers projets d'entreposage [70], elles ne font pas l'objet d'étude de notre thèse.

1.3.2.1 Problématique d'intégration

Un système d'intégration a pour but de fournir un schéma cohérent des données provenant de multiples sources d'informations autonomes, réparties et hétérogènes, de manière à faciliter

aux utilisateurs l'accès et l'interrogation de ces données comme s'ils accédaient à une seule source de données. Un système d'intégration comporte deux parties (figure 1.17) [16] : une partie *externe* représentée par les utilisateurs du système intégré ; et une partie *interne* correspondant aux sources de données et à une interface (schéma global) à travers laquelle les utilisateurs accèdent aux sources. Un système d'intégration est par conséquent, formellement défini par le triplet <G,S,M> [120] où G représente le schéma global du système d'intégration, S représente l'ensemble des sources à intégrer et M décrit différentes assertions de mapping entre le schéma global et les sources. Un ED est dans ce cas vu comme un système d'intégration de données matérialisant les données via un processus ETL.

1.3.2.2 Hétérogénéité des données

L'hétérogénéité des sources de données est à l'origine de la complexité de la tâche d'intégration. Cette hétérogénéité peut être de deux natures : hétérogénéité *structurelle* (ou *syntaxique*) et hétérogénéité *sémantique*. L'intégration des données implique l'identification des conflits syntaxiques et sémantiques et ensuite leur résolution.

1.3.2.2.1 Hétérogénéité structurelle

L'hétérogénéité structurelle provient du fait que les sources de données peuvent avoir différentes structures ou différents formats de stockage. Les différences structurelles peuvent être regroupées dans différentes catégories [187] :

- *Choix du type de données* : ces conflits se posent lorsqu'on utilise des types de données différents pour la même information. Par exemple, dans le domaine des transactions commerciales, la quantité d'un produit est représentée par un réel dans une source S1 et par une chaîne de caractère dans une autre source S2.
- *Choix du nombre de constructeurs* : ces conflits se présentent lorsque le nombre de constructeurs modélisant une information est différent d'une source à une autre. Par exemple, l'attribut nom d'un client est modélisé par un seul attribut servant à stocker le nom et le prénom d'un client dans une source S1, alors que deux attributs sont utilisés dans une autre source 2.
- *Choix des informations représentées* : ces conflits se posent lorsqu'une information est représentée dans des sources alors qu'elle ne l'est pas dans d'autres. Par exemple, l'adresse d'un client n'est pas connue pour tous les clients d'une source S1, alors que c'est une donnée obligatoire dans une source S2.

1.3.2.2.2 Hétérogénéité sémantique

L'hétérogénéité sémantique représente une problématique plus difficile à gérer. Elle provient du fait que les sources sont conçues par différents concepteurs qui ont des objectifs applicatifs différents et ne partagent donc pas forcément la même sémantique des concepts [16]. Les conflits sémantiques peuvent être de différentes natures. *Goh et al.* [67] distinguent les trois types de conflits sémantiques suivants (figure 1.18) :

- *Conflits de noms* : les conflits de noms se produisent lorsqu'on utilise soit des noms différents pour le même concept ou propriété (synonyme), ou plus rarement des noms identiques pour des concepts différents (homonyme). Un simple exemple se produit si l'on trouve le

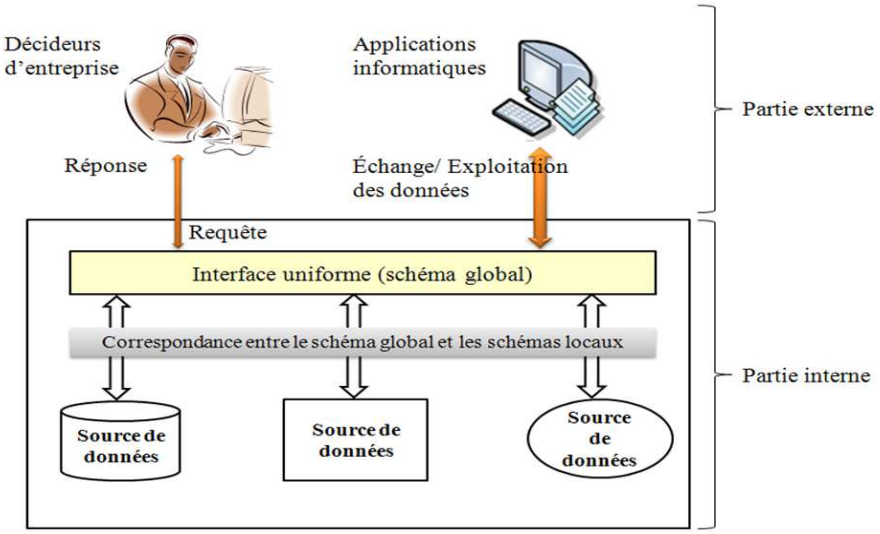

FIGURE 1.17 – Système d'intégration des données

concept Produit dans la source S1 et Article dans la source S2, alors que les deux concepts portent le même sens dans les deux sources. Dans un autre cas, on retrouve l'attribut Prix dans les deux sources, mais qui signifie le prix de vente d'un produit dans la source S1, et le prix de production d'un produit dans la source S2.

- *Conflits de contextes* : les conflits de contextes se produisent lorsque des concepts semblent avoir la même signification mais ils sont évalués dans différents contextes. Par exemple, la propriété Prix ne s'applique que pour les produits neufs dans la source S1, alors qu'elle est appliquée pour tous les produits dans la source S2.
- *Conflits de mesures* : les conflits de mesures ou de valeurs se trouvent dans le cas où des unités de mesure différentes ont été utilisées pour mesurer certaines propriétés de certains concepts. Par exemple, la valeur de l'attribut Prix d'un produit est calculée en dinars dans la source S1 et en euros dans la source S2.

1.3.3 Entrepôt de données vu comme une base de données

La deuxième génération des projets d'entreposage a reconnu l'importance de deux phases de conception : l'analyse des besoins et la modélisation conceptuelle. Même si ces phases étaient effectuées dans la première génération des projets d'entreposage, probablement de manière implicite, elles n'ont été reconnues comme phases à part entière qu'à partir de la fin des années 90. *Ralph Kimball* [107], un acteur important du domaine des ED, a d'abord fourni une vision de conception de l'ED à partir des besoins des utilisateurs. Sa proposition a été suivie par plusieurs autres études proposant des méthodes de conception orientées besoins. Nous les analyserons dans le chapitre suivant de l'état de l'art. La phase de modélisation conceptuelle a ensuite été reconnue comme phase indispensable [70]. Le cycle de conception d'un ED inclut ainsi les phases suivantes : analyse des besoins, modélisation conceptuelle, modélisation logique, processus ETL et modélisation physique.

Ce cycle de conception donne une nouvelle vision du développement des ED, similaire à la

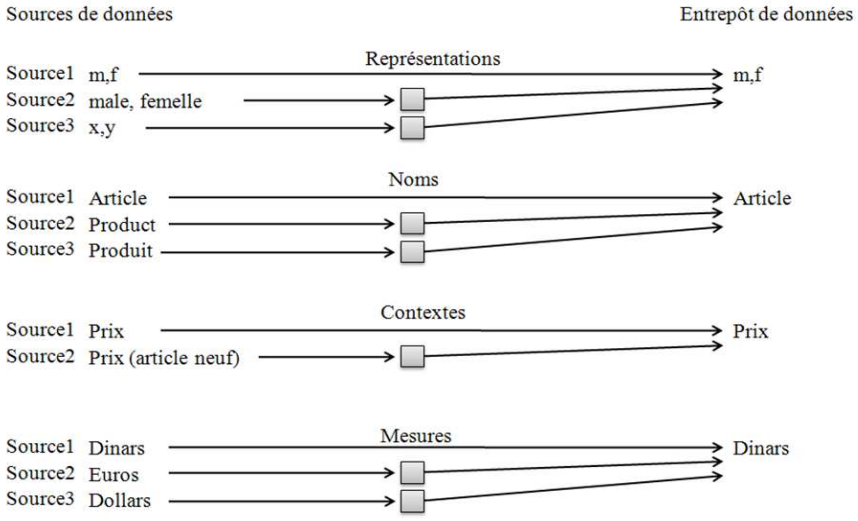

FIGURE 1.18 – Hétérogénéité sémantique des donnée

conception des BD. Cette vision est partagée par plusieurs auteurs qui considèrent les ED comme des BD, ayant une charge de requêtes particulière (nécessitant généralement des opérations de lecture et de rafraîchissement) [86]. Les BD et également les ED visent le même objectif de consolidation des données : les BD en vue de faciliter la gestion des données, les ED en vue de faciliter l'analyse des données. Cette consolidation des données peut être d'une plus grande envergure dans certains projets d'ED où il faut intégrer des sources multiples, parfois géographiquement distribuées à travers le monde. Le tableau 1.1 récapitule les principales différences entre les BD opérationnelles et les ED [14]. Nous décrivons ensuite la définition et l'objectif de chaque phase du cycle de conception.

	Base de données	Entrepôt de données
But	Exécution d'un processus métier	Evaluation d'un processus métier
Interaction avec l'utilisateur	Insertion, Modification, Interrogation, Suppression	Interrogation
Données	Courantes	Courantes et Historiques
Usage	Support de l'opération de l'entreprise	Support de l'analyse de l'entreprise
Requêtes	Simples, Prédéterminées	Complexes, Ad-hoc
Principe de conception	Troisième forme normale	Conception multidimensionnelle

TABLE 1.1 – Comparaison Base de données et Entrepôt de données

1.3.3.1 Définition des besoins

La définition des besoins joue un rôle clé dans tout projet logiciel en permettant de réduire significativement le risque d'échec du projet [158]. La définition des besoins pour les applications d'ED est définie comme le processus de développement des besoins selon un processus itératif

et coopératif d'analyse du problème, de documentation des observations résultantes dans divers formats de représentation et de vérification des résultats obtenus [30]. Deux types de besoins sont distingués : les besoins *fonctionnels* et les besoins *non fonctionnels*. Un besoin non fonctionnel est défini comme un attribut ou une contrainte du système comme la flexibilité, la performance, la sécurité, etc. [66]. Deux méthodes sont utilisées pour collecter les besoins : une collecte orientée source et une collecte orientée utilisateur. Les approches considérant une phase de définition des besoins des utilisateurs lors de la conception de l'ED sont appelées *approches orientées besoins* ou approches *descendantes*. Les *approches orientées sources* ou *ascendantes* se limitent à identifier les besoins de l'ED à partir de l'ensemble des données disponibles au niveau des sources. La collecte des besoins auprès des utilisateurs permet de se concentrer à fournir un modèle répondant à ce qui est exigé plutôt que ce qui est disponible. La définition des besoins dans un projet d'ED passe par les activités suivantes [214] :

- *Planning de gestion des besoins* : consiste à définir les objectifs du projet par les utilisateurs et les concepteurs, à définir les règles d'intégration des sources de données et à établir un planning de gestion des besoins du projet (contraintes de temps, délimiter les frontières du projet, etc).
- *Spécification des besoins* : s'effectue par un processus itératif d'acquisition ou collecte des besoins [13], ensuite de représentation et spécification des besoins.
- *Validation des besoins* : consiste à valider les modèles initiaux construits lors de l'étape de spécification des besoins, avec les utilisateurs ainsi qu'avec les sources de données existantes. La validation des besoins est menée par l'équipe de développement, les utilisateurs et les experts du domaine. L'expérience a montré en effet que même après l'implémentation de l'ED, un retour vers la phase d'identification des besoins est possible [13].
- *Suivi et gestion de l'évolution des besoins* : la gestion doit être effectuée à deux niveaux : une gestion de l'évolution des besoins des utilisateurs, et une gestion de l'évolution de l'architecture des sources afin d'étudier l'impact d'éventuels changements sur les modèles suivants (conceptuel, logique et physique).

1.3.3.2 Modélisation conceptuelle

La deuxième phase de modélisation conceptuelle consiste à définir le schéma conceptuel de l'ED annoté par les concepts multidimensionnels. Plusieurs méthodes ont été proposées pour ce faire. Leur but est de fournir un modèle conceptuel de l'ED fournissant une représentation abstraite de la situation en cours d'étude indépendamment de toute contrainte d'implémentation technique. Ce modèle conceptuel peut être défini à partir du modèle de besoins. Il peut aussi être représenté comme une vue intégrée des données, définie au niveau conceptuel si l'approche de conception est strictement orientée sources. Un modèle conceptuel est caractérisé par le domaine concerné, le formalisme utilisé pour modéliser le schéma conceptuel, et le point de vue correspondant aux besoins des utilisateurs [58]. Le modèle conceptuel obtenu doit se conformer à la modélisation multidimensionnelle adaptée aux modèles d'ED permettant d'organiser les données entreposées de manière à faciliter leur analyse décisionnelle.

Produit	Temps	Trimestre 1			Trimestre 2			Trimestre 3			Trimestre 4			Total		
	Site	N	S	Total	N	S	Total	N	S	Total	N	S	Total	N	S	Total
Produits laitiers		12	34	46	22	36	58	24	37	61	33	55	88	91	162	253
Viennoiseries		29	66	95	44	50	94	56	55	111	44	39	83	173	210	383
Viandes		55	34	89	69	27	96	31	26	57	68	70	138	223	157	380
Total		96	134	230	135	113	248	111	118	229	145	114	309	487	529	1016

FIGURE 1.19 – Représentation du cube multidimensionnel selon MOLAP

1.3.3.3 Modélisation logique

La modélisation logique de l'ED passe par la gestion des trois principaux aspects [108] :

- *Le suivi des données* : consiste à concevoir le modèle logique cible de l'ED répondant aux besoins des utilisateurs, tout en tenant compte des spécificités des données des sources. La conception du modèle logique passe par les tâches suivantes : (1) la traduction du modèle conceptuel multidimensionnel en un modèle logique. (2) L'identification des sources de données candidates pour alimenter le schéma de l'ED et pour préparer la prochaine étape ETL. (3) La sélection des sources candidates pour alimenter le modèle de l'ED, (4) le mapping entre les données des sources et le schéma logique cible de l'ED et enfin (5) l'estimation de la charge de l'ED.
- *Le suivi technologique* : l'application décisionnelle nécessite l'intégration de nombreuses technologies et de gestionnaires de stockage des données. Le suivi technologie consiste à fournir une vue abstraite de l'architecture technique de l'ED.
- *Le suivi de l'application décisionnelle* : consiste en l'identification de l'application décisionnelle supportant l'ED et passe par le développement des rapports normalisés, des requêtes paramétrées, des tableaux de bord, des modèles analytiques, des applications d'exploration de données ainsi que les interfaces associées à la navigation.

En sortie, la phase de modélisation logique doit fournir le modèle logique de l'ED, adapté aux particularités du modèle d'implémentation logique choisi. Trois principales représentations logiques sont distinguées : ROLAP, MOLAP ou HOLAP, que nous détaillons dans ce qui suit.

1.3.3.3.1 L'approche MOLAP

L'approche MOLAP (Multidimensional On-Line Analytical Processing) implémente le cube sous forme d'un tableau multidimensionnel, qui sera ensuite implémenté dans un SGBD multidimensionnel. Chaque dimension du tableau représente une dimension du cube. Les données de chaque cellule sont stockées. Le principal avantage de l'approche MOLAP est sa performance en temps d'accès. Par exemple, le cube multidimensionnel des ventes de la figure 1.15 peut être représenté par le tableau multidimensionnel représenté dans la figure 1.19.

1.3.3.3.2 L'approche ROLAP

L'approche ROLAP (Relational On-Line Analytical Processing) représente le modèle de l'ED

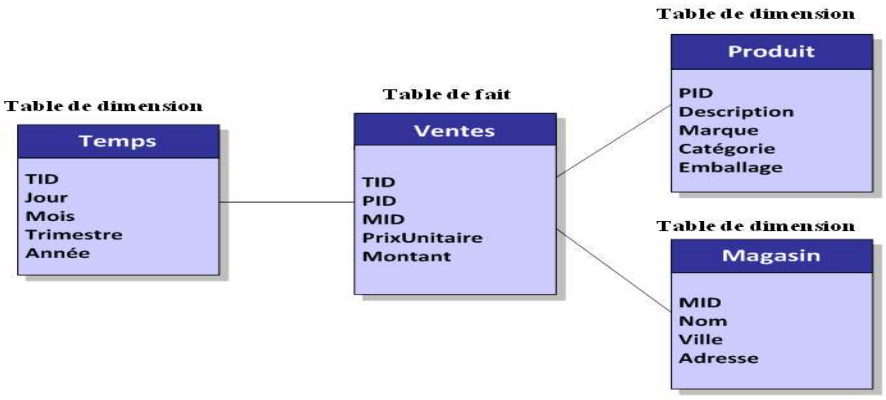

FIGURE 1.20 – Représentation du cube multidimensionnel selon ROLAP

selon une représentation relationnelle, qui sera ensuite implémenté en utilisant un SGBD relationnel. Chaque dimension du cube est représentée sous forme d'une table appelée table de dimension. Chaque fait est représenté par une table de fait. Les mesures sont stockées dans les tables de faits qui contiennent les valeurs des mesures et les clés vers les tables de dimensions. Par exemple, le cube multidimensionnel des ventes de la figure 1.15 peut être représenté par les tables de la figure 1.20.

Le principal inconvénient des systèmes ROLAP est qu'ils peuvent présenter un temps de réponse aux requêtes élevé. Leurs principaux avantages sont : (1) l'exploitation des capacités d'un standard bien établi et maîtrisé (le modèle relationnel). (2) Le stockage de grandes quantités de données. Trois schémas sont utilisés pour modéliser les systèmes ROLAP : (1) le schéma en étoile, (2) le schéma en flocon de neige et (3) le schéma en constellation.

Le schéma en étoile : chaque dimension du cube est représentée par une table de dimension et les mesures par une table de faits qui référence les tables de dimension en utilisant une clé étrangère pour chacune d'elles (figure 1.20). La table de faits est normalisée, les tables de dimension sont généralement dénormalisées.

Le schéma en flocon de neige : le schéma en flocon de neige est une extension du schéma en étoile. Dans un schéma en étoile, les informations associées à une hiérarchie de dimension, sont représentées dans une seule table, même si les différents niveaux de la hiérarchie ont des propriétés différentes (figure 1.21). Le schéma en flocon est le résultat de la décomposition d'une ou plusieurs dimensions en plusieurs niveaux formant une hiérarchie. Les tables de dimensions sont ainsi éclatées en plusieurs tables, ce qui peut être vu come une normalisation des tables de dimensions. La table de faits reste inchangée. Ce type de schéma offre une meilleure visualisation et compréhension des données, mais peut altérer les performances de l'entrepôt lors de son utilisation. En effet, une requête nécessitera plusieurs jointures ce qui augmente son temps de réponse.

Le schéma en constellation : les schémas en constellations sont des schémas où plusieurs modèles dimensionnels se partagent les mêmes dimensions. Les tables de dimensions partagées par plusieurs tables de fait doivent être exactement les mêmes.

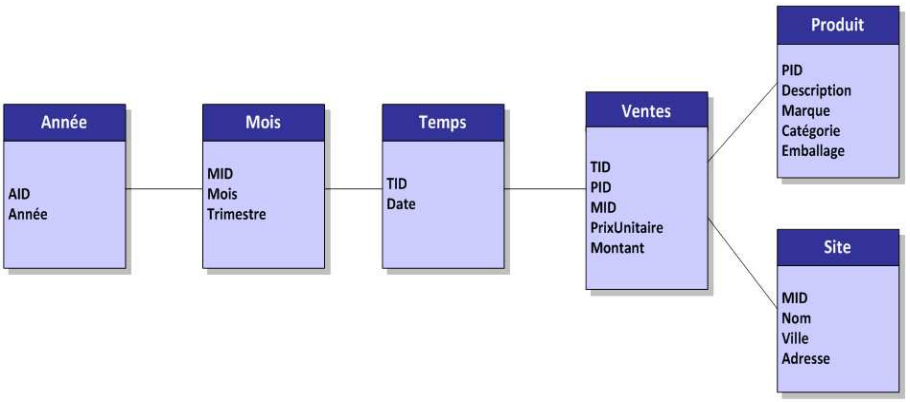

FIGURE 1.21 – Exemple d'un schéma en flocon de neige

1.3.3.3.3 L'approche HOLAP

L'approche HOLAP (Hybrid On-Line Analytical Processing) représente les données fréquemment utilisées (généralement les données agrégées) dans un tableau multidimensionnel, et représente les données non fréquemment utilisées dans un schéma relationnel. Ceci afin de bénéficier des avantages des deux approches citées précédemment. La séparation des données doit être transparente à l'utilisateur final.

Le tableau 1.2 propose une comparaison entre les deux approches ROLAP et MOLAP [14, 74] :

Stockage	Avantages	Inconvénients
ROLAP	Standard, technologie prouvée et familière	Lenteur du traitement des requêtes
	Scalable (capacité d'expansion élevée)	Conception basée sur les diagrammes EA pas toujours appropriée pour les systèmes d'aide à la décision
MOLAP	Modèle multidimensionnel (calcul automatique des agrégations des données)	Redondance des données
	Traitement de requête spécialisé et rapide	Non scalable
	Techniques d'indexation spécialisées	Consommation de l'espace lorsque les données sont éparses
HOLAP	Accès rapide pour différents niveaux d'agrégation	Système complexe (un serveur HOLAP doit supporter des outils MOLAP et ROLAP)
	Stockage compact des agrégations	Charge importante (entre le stockage et les techniques d'optimisation ROLAP et les outils MOLAP)

TABLE 1.2 – Ccomparaison ROLAP, MOLAP et HOLAP

La représentation ROLAP étant la plus utilisée, nous utilisons la terminologie relative aux modèles relationnels (table, clé, colonnes, tuples, etc) pour expliciter les étapes des phases suivantes.

1.3.3.4 Phase ETL (Extract-Transform-Load)

Comme son nom l'indique, cette phase consiste à extraire les données à partir des sources sélectionnées et à effectuer les transformations nécessaires pour assurer le chargement des données des sources au niveau du schéma cible de l'entrepôt. Les données sont extraites à partir des sources de données hétérogènes. Elles sont ensuite propagées dans une zone de stockage temporaire : Data Staging Area (DSA), où auront lieu leur transformation, homogénéisation et nettoyage. Enfin, les données sont chargées dans l'ED cible [202]. Il existe actuellement une panoplie d'outils ETL qui ont été proposés par la communauté industrielle, comme *Microsoft Data Transformation Services* (DTS), *Oracle Warehouse Builder* (OWB), *IBM Data Warehouse Center*, *Informatica PowerCenter*, *Talend open studio*, *Pentaho Data Integration* (PDI). D'autres outils ETL académiques ont été proposés comme le système *AJAX* [61], l'outil *ARKTOS* [204], l'outil *HumMer* [140] et le système *Potter's Wheel* [157].

1.3.3.5 Modélisation physique

Cette étape consiste à implémenter physiquement le modèle logique de l'ED et à spécifier les techniques et schémas d'optimisation de l'entrepôt, accompagné d'une spécification détaillée des caractéristiques physiques de l'ED (les types de données, la segmentation des tables, les paramètres de stockage, la spécification des clés des tables, la gestion des instances). Le choix et l'application des techniques d'optimisation se fait également lors de la phase physique. Les techniques d'optimisation peuvent être redondantes ou non redondantes [14]. Les techniques redondantes sont des techniques d'optimisation des requêtes qui nécessitent un stockage physique des données et une maintenance. On distingue : les index, les vues matérialisées et la fragmentation verticale. Les techniques non redondantes sont des techniques d'optimisation des requêtes qui ne nécessitent pas un stockage physiques des données. On distingue : la fragmentation horizontale primaire et dérivée. Cette phase de modélisation physique aboutit au déploiement de l'ED faisant converger les trois aspects étudiés lors de la phase de modélisation logique, réunissant la technologie, le modèle de données et les applications décisionnelles développées. Cette phase nécessite l'installation de la plateforme de déploiement et des outils nécessaires permettant aux utilisateurs finaux d'accéder à l'ED, la gestion des aspects de sécurité, la vérification de la qualité des données, le développement de l'application décisionnelle pour les utilisateurs finaux, la formation des utilisateurs et l'établissement de la documentation nécessaire.

1.4 Conclusion

Nous avons exposé dans ce premier chapitre un historique retraçant les principales évolutions de la conception des BD ayant mené à l'établissement de leur cycle de conception. Nous avons distingué trois types d'évolutions : une évolution *verticale* donnant lieu aux trois niveaux de conception de l'architecture ANSI/SPARC, une évolution *interne* détaillant les étapes, les entrées/sorties et les acteurs intervenant dans chaque phase de conception, et une évolution *horizontale* étendant chaque phase du cycle de conception par de nouveaux modèles de données conceptuels, logiques et physiques et par de nouvelles plateformes de déploiement.

La stabilité du cycle de conception a mené au développement de plusieurs applications de BD

et donc à la génération de volumes importants de données qu'il fallait analyser. Les ED ont été la solution adoptée pour permettre l'intégration des données issues de sources hétérogènes afin d'y effectuer différents types d'analyses. Le cycle de conception des ED est également passé par plusieurs évolutions pour converger vers un cycle incluant les cinq principales phases suivantes : la définition des besoins, la modélisation conceptuelle, la modélisation logique, le processus ETL et la modélisation physique. Nous avons relevé deux principaux constats :

- En premier lieu, nous constatons que le cycle de conception des ED n'a fait que consolider le cycle de conception établi pour les BD, en l'enrichissant par de nouveaux modèles conceptuels, logiques et physiques.
- En second lieu, nous avons remarqué que la diversification des modèles de données au sein de ce cyle répond généralement à une nécessité d'incorporer davantage de sémantique au sein de la BD/ED. Les BDBO ont présenté une solution pour permettre la représentation de la sémantique des *données* de la base sous forme d'une ontologie locale. Malgré cette avancée, nous constatons que seules les données de l'application sont stockées au sein de la BD/ED. La première phase de définition des besoins permet de fournir l'ensemble des données (conceptuel, logique et physique) de la base, et l'ensemble de ses traitements.

Nous proposons ainsi dans notre première proposition de revisiter le cycle de conception afin de faire persister ce premier modèle source des autres modèles : *le modèle de besoins*, ceci dans le même objectif d'incorporation de la sémantique de l'application dans l'ED.

Notre deuxième proposition se concentrera sur la phase ETL du cycle de conception. Nous avons remarqué que le cycle de conception des ED s'est construit en deux principales générations : la première génération des ED a considéré l'ED davantage comme un système d'intégration où le processus d'intégration s'effectue lors de la phase ETL. Une deuxième génération qui a inclu les deux phases de modélisation conceptuelle et de définition des besoins. La phase ETL reste placée suite à la phase de modélisation logique. Cette position restreint le déploiement de l'ED à un choix unique qui suit le modèle logique et physique adoptés. Nous verrons dans le chapitre suivant de l'état de l'art, que plusieurs travaux tenteront de modéliser cette phase ETL aux différents niveaux conceptuel, logique et physique. Notre contribution se situe dans la continuité de ces travaux et consiste à revisiter une seconde fois le cycle de conception en revoyant l'ordre d'exécution du processus ETL. Nous estimons que la phase ETL doit être conçue dès la phase conceptuelle, plus précisément au niveau ontologique. Cette deuxième proposition répond à notre objectif de consolidation du cycle de conception en permettant un *déploiement à la carte* de l'ED sur les diverses plateformes disponibles.

Nous présenterons dans le prochain chapitre une revue de littérature des différentes méthodes de conception proposées et leurs contributions au sein de chaque phase du cycle de conception.

Chapitre 2

États de l'Art sur la conception des entrepôts de données

Sommaire

2.1 Introduction

Un retour sur l'historique du cycle de conception des projets d'entreposage nous a permis de mieux analyser les différentes visions de conception des ED. Cet historique remonte aux années 90, peu après la parution du concept d'ED. Le concept d'entrepôt de données étant issue de l'industrie, la première génération des approches de conception considéraient l'ED comme un système d'intégration matérialisé par des concepts multidimensionnels. Les aspects d'optimisation ont été largement étudiés dans ces travaux. La prise en compte des besoins des utilisateurs a rapidement été évoquée par des propositions de méthodes de conception qui ont montré l'importance des besoins pour le succès du projet d'entreposage. Deux principales catégories de méthodes ont ainsi été proposées : une première catégorie *orientée sources* se basant uniquement sur les sources de données et une deuxième catégorie *orientée besoins* reconnaissant la nécessité d'inclure les besoins des utilisateurs de l'ED lors de sa conception. Des méthodes dites *mixtes* (ou hybrides) ont ensuite proposé de définir le schéma de l'ED à partir des sources et des besoins des utilisateurs. La mise en place d'une phase de modélisation conceptuelle pour le développement des ED s'est imposée par la suite afin de de fournir un schéma défini à un niveau d'abstraction indépendant de toute contrainte d'implémentation. Le cycle de conception des ED reconnu actuellement inclut les phases de définition des besoins, modélisation conceptuelle, modélisation logique, processus ETL et modélisation physique. La conception des ED continue de faire l'objet de recherches actives. Plusieurs méthodes ont été proposées, apportant une contribution dans une ou plusieurs phases du cycle de conception des ED. Nous remarquons que les différentes propositions se divisent en deux parties : une catégorie de propositions portant sur une ou plusieurs phases de conception du *schéma* de l'ED : conceptuelle, logique et physique. Une deuxième catégorie de propositions porte sur la phase de *conception ETL*, et permet donc d'alimenter le schéma de l'ED. Quelques travaux récents, très peu nombreux, proposent une démarche conjointe de conception impliquant les phases de conception et d'ETL. Nous proposons dans ce chapitre d'étudier les différents travaux proposés selon ces deux catégories de propositions. La première section du chapitre porte sur l'étude des travaux de conception du schéma de l'ED. La deuxième section porte sur l'étude des travaux de conception ETL de l'ED. Dans chacune des sections, nous présenterons une synthèse des travaux retenus, leurs principales contributions et une classification de ces travaux selon un ensemble de critères retenus. Ces critères sont choisis pour leur pertinence pour nos contributions. Nous ressortons les principales difficultés et limites liées à ces travaux, et nous mettons en avant les problèmes sur lesquels nous allons tenter d'apporter des solutions. Les travaux que nous avons retenus dans nos états de l'art sont les travaux les plus référencés dans la littérature, et aussi les travaux récents proposés ces dernières années. Ces travaux sont présentés par ordre d'apparition. D'autres travaux moins référencés sont uniquement cités dans la classification que nous fournissons pour analyser les propositions de l'état de l'art.

2.2 Conception du schéma de l'ED

2.2.1 Etat de l'art des travaux de conception

Nous présentons dans cette section les principaux travaux proposant des méthodes de conception du schéma de l'ED, couvrant une ou plusieurs des phases suivantes : définition des besoins,

modélisation conceptuelle, modélisation logique et modélisation physique.

Kimball [107] propose une des premières approches de conception d'ED. L'approche n'est pas formalisée mais peut être vue comme un guide détaillé permettant d'identifier les concepts multidimensionnels donnant lieu au schéma de l'ED. L'approche globale consiste à identifier les magasins de données considérés comme des collections de données représentant des faits et de lister les dimensions possibles pour chaque magasin. Le schéma en étoile de l'ED est construit en analysant les besoins identifiés à partir des processus organisationnels, selon les étapes suivantes : (1) sélectionner les processus organisationnels à analyser : un processus est défini comme une activité habituelle effectuée dans l'organisation qui est généralement appuyée par une source de données d'un système transactionnel. (2) Déterminer la granularité des données à prendre en compte, c'est à dire le niveau de détail des données à analyser qui doit être représenté dans le schéma en étoile. Il est recommandé de considérer les données de granularité très fine (données atomiques) qui sont très détaillées et ne peuvent être divisées. (3) Choisir les dimensions ou axes d'analyse possibles, en répondant à la question : comment les analystes décrivent-ils les données résultantes des processus organisationnels sélectionnés ? (4) Déterminer les tables de faits représentant l'objet de l'analyse pour chaque processus en répondant à la question : que doit-on mesurer ?.

Cabibbo et al. [33] proposent de générer le schéma logique multidimensionnel d'un ED à partir d'un ensemble de schémas E/A des sources. Les deux schémas logiques (relationnel et multidimensionnel) sont générés. Cette méthode comprend quatre principales étapes : (1) l'identification des faits à partir des besoins des utilisateurs, (2) l'identification des dimensions à partir des besoins des utilisateurs. Ces deux premières étapes peuvent être effectuées en parallèle. L'identification des faits et des dimensions se fait manuellement par le concepteur. Chaque fait est représenté par une entité. (3) L'analyse des sources permet d'identifier les dimensions et les niveaux de dimensions complémentaires. (4) Le schéma multidimensionnel est généré sous forme d'un graphe. Quelques règles sont définies pour sa représentation dans un schéma relationnel ou sous forme d'un tableau multidimensionnel.

Golfarelli et al. [70] proposent une méthode semi automatisée très souvent référencée par d'autres travaux. La méthode permet de concevoir un modèle conceptuel d'un ED à partir des schémas E/A décrivant les BD sources. Le modèle conceptuel défini est baptisé Dimensional Fact model (modèle DF). La méthode a été réactualisée dans [68]. Le modèle conceptuel proposé consiste en un ensemble de schémas DF représentés graphiquement sous forme d'un arbre dont la racine est le fait. La méthode comprend six étapes : (1) l'analyse des sources de données pour générer le schéma conceptuel décrivant ces sources. (2) La collecte et l'analyse des besoins des utilisateurs. (3) La construction du schéma conceptuel multidimensionnel (représenté par un modèle DF). Ce modèle est généré en commençant par définir les faits. Les entités et relations ayant subi le plus de modifications et de mises à jour sont considérées comme des candidats potentiels. Un fait peut être représenté par une ou plusieurs entités. Pour chaque fait défini, la deuxième étape consiste à construire l'arbre des attributs (les attributs des entités sélectionnées comme fait), dont la racine est l'identifiant du fait et les nœuds ses attributs. L'arbre est affiné en éliminant les attributs les moins intéressants pour l'utilisateur. Les dimensions sont définies et correspondent aux nœuds racines de l'arbre. Les mesures des faits sont définies en évaluant des fonctions d'agrégation (somme/moyenne/maximum/minimum) sur les attributs numériques

de l'arbre. Les hiérarchies de dimensions sont finalement définies en identifiant les relations un à plusieurs[3] entre chaque nœud "dimension" et ses descendants. (4) La définition du modèle logique de l'ED se fait en traduisant chaque fait et chaque dimension identifiés en une table relationnelle. (5) L'étape suivante consiste à traduire le modèle logique au niveau physique. Les auteurs fournissent quelques directives pour implémenter le modèle dans un outil ROLAP, et pour définir quelques structures d'optimisation comme les index. (6) La dernière étape estime la charge de requêtes de l'ED dans le but de valider le schéma conceptuel multidimensionnel généré mais aucune directive n'est fournie.

Boehnlein et al. [24] dérivent un schéma en étoile de l'ED à partir des schémas E/A des BD sources. Ces schémas sont d'abord restructurés pour permettre de visualiser les dépendances entre les objets. Les mesures sont identifiées à partir des buts émis par les utilisateurs. Les évènements enregistrés sont des candidats potentiels pour représenter des mesures. Ces dernières sont ensuite mises en correspondance avec un ou plusieurs objets dans les schémas des sources. Ces mesures donneront lieu à des faits. Les hiérarchies de dimensions sont construites en sélectionnant les entités liées aux entités représentant les mesures par des relations de dépendances fonctionnelles directes ou transitives. Un schéma en étoile ou en flocon de neige est dérivé selon que les hiérarchies de dimensions sont regroupées ou pas.

Husemann et al. [88] présentent une méthode permettant de générer un schéma conceptuel d'un ED à partir des BD existantes. Les auteurs proposent un processus de conception d'un ED similaire à celui des BD classiques, selon les quatre étapes suivantes : définition des besoins, modélisation conceptuelle, modélisation logique et modélisation physique. L'analyse des besoins consiste à analyser les attributs pertinents à partir des schémas sources pour spécifier leur rôle (fait, mesure ou dimension). Ces attributs sont décrits dans un tableau similaire au dictionnaire des données pour les BD. La modélisation conceptuelle consiste à définir le schéma conceptuel multidimensionnel à partir du dictionnaire de données. Un fait représente un élément d'informations atomique. Les mesures sont représentées par des éléments quantifiant le fait, et les dimensions comme des éléments le qualifiant. Les hiérarchies de dimension sont définies en déterminant les dépendances fonctionnelles entre les différents niveaux de la hiérarchie. Chaque hiérarchie de dimension doit avoir un niveau atomique, et chaque niveau de dimension est identifié par un attribut. Les dimensions générées doivent être orthogonales. Ces contraintes assurent au schéma multidimensionnel généré de respecter la forme normale multidimensionnelle définie par Lehner et al. [118].

Moody et al. [138] proposent de développer le schéma multidimensionnel de l'ED à partir des schémas E/A en classant les différentes entités en trois classes : (1) les entités transactionnelles qui représentent les entités enregistrant des détails concernant des évènements particuliers de l'entreprise, (2) les entités composantes : ce sont les entités qui décrivent le détail des entités transactionnelles, et qui sont directement liées aux entités transactionnelles par une relation un à plusieurs, et (3) les entités de classification : qui correspondent à toute séquence d'entités liées entre elles par des relations un à plusieurs. Les auteurs soulignent la nécessité d'utiliser les besoins des utilisateurs afin de déterminer les entités transactionnelles. Les entités transactionnelles représenteront les entités *fait*, les entités composantes représenteront les entités *dimension*, et les

3. Dans une relation un à plusieurs (one to many relationship), le nombre maximum d'instances participant à la relation est de un dans un sens et supérieur à un dans l'autre.

entités de classification représenteront les *hiérarchies* de dimension. Ces dernières sont ensuite raffinées en déterminant le niveau de granularité des entités dimensions. Les auteurs définissent ensuite deux principaux opérateurs : l'opérateur Collapse permettant de regrouper les niveaux de dimensions aux niveaux inférieurs et l'opérateur Aggregation qui est appliqué aux entités transactionnelles pour créer une nouvelle entité contenant les données résumées. Sur la base de ces deux opérateurs, les auteurs définissent des règles pour dériver le schéma logique de l'ED. Cinq types de schémas sont identifiés parmi lesquels se trouvent le schémas en étoile et le schéma en flocon de neige.

Bonifati et al. [25] proposent une méthode de conception mixte permettant la définition du schéma de l'ED. Un premier ensemble de schémas multidimensionnels est dérivé à partir des buts organisationnels, en adaptant l'approche Goal/Question/Metric (GQM). GQM est une approche permettant de fournir des métriques pour mesurer des programmes. Cette approche a été adaptée pour identifier les buts d'une organisation et les analyser afin d'en extraire des informations pertinentes. Les buts sont d'abord collectés à travers les interviews avec les utilisateurs et structurés selon un format détaillant chaque but. Quelques directives générales sont fournies afin de générer des schémas multidimensionnels à partir de l'ensemble de buts. Un deuxième ensemble de schémas multidimensionnels (sous forme de graphes) sont dérivés automatiquement à partir de schémas E/A des BD sources. Un algorithme est défini qui identifie les faits comme les entités ayant des attributs numériques. Ces entités représentent les nœuds centraux du graphe. Les dimensions sont les entités liées aux entités Fait par une relation 'un à un' ou une relation 'un à plusieurs'. Un algorithme est fourni afin de traduire chaque graphe en un schéma en flocon de neige. Une dernière étape consiste à intégrer les deux ensembles de schémas multidimensionnels (les schémas obtenus à partir des buts et les schémas obtenus à partir des sources) en un schéma unifié. Ce mapping est effectué manuellement par le concepteur et nécessite la mise en correspondance de l'ensemble des schémas à un référent terminologique unique (un dictionnaire par exemple). Les deux ensembles de schémas sont comparés un à un. Les schémas sont intégrés s'ils possèdent la même entité de *fait*.

Phipps et al. [150] proposent de définir le schéma conceptuel multidimensionnel de l'ED. Un premier modèle multidimensionnel est généré à partir des schémas relationnels des sources de données. Les tables relationnelles ayant des attributs numériques représentent des faits, leurs attributs numériques représentent des mesures, et les autres attributs (attributs non numériques, non clé et ne représentant pas une date) représentent les dimensions de ce fait. Une dimension Temps est associée à toutes les entités faits. Les niveaux de dimension sont définis en identifiant de manière récursive les tables liées aux tables dimensions par une dépendance fonctionnelle. Un ensemble de schémas multidimensionnels est généré par cet algorithme. Ces schémas sont ensuite validés et raffinés par les besoins des utilisateurs formalisés par des requêtes SQL. La pertinence des schémas est évaluée selon les règles suivantes : (1) si un schéma ne contient pas de tables se trouvant dans la clause From d'une requête, alors ce schéma est considéré comme non pertinent pour cette requête. (2) Un schéma est considéré comme pertinent si les attributs numériques de son fait se trouvent dans la clause Select.

Vrdoljak et al. [209] présentent une approche semi-automatique pour la conception logique d'un ED à partir des schémas XML (considérés comme les sources de données). Cette approche comprend cinq étapes : (1) le prétraitement et la simplification des schémas XML afin d'éliminer

les relations redondantes. (2) La transformation de chaque schéma XML en un graphe. Les dépendances fonctionnelles sont explicitement représentées dans le graphe. Les nœuds du graphe qui ne stockent pas de valeur sont exclus. (3) Les faits sont ensuite identifiés parmi les nœuds ou les arcs du graphe. (4) Un graphe de dépendance est généré pour chaque fait, en identifiant les relations un à plusieurs ou plusieurs à plusieurs. Ces dernières doivent être validées par le concepteur. Ce graphe de dépendance permet de définir les dimensions et leurs hiérarchies. (5) Le schéma logique de l'ED est enfin dérivé à partir des faits et des dimensions identifiées.

Winter et al. [213] présentent une méthode globale sous forme de directives à suivre pour la conception d'un ED. Cette méthode a été développée à partir de l'analyse de plusieurs projets réels d'entrepôts de données dans des entreprises. Le processus de conception doit être itératif et comprend quatre principales étapes : (1) l'analyse des sources de données. (2) L'analyse des besoins des utilisateurs afin de valider les données pertinentes à partir des sources. (3) Les informations issues de l'analyse des sources et des besoins doivent être détaillées et mises en correspondances. (4) La génération du schéma multidimensionnel qui doit être validé.

Jensen et al. [92] proposent de dériver le schéma logique de l'ED à partir de schémas relationnels des sources. L'approche proposée a pour première étape de consulter le catalogue des bases de données sources. La structure de chaque BD est enrichie par l'explicitation des dépendances fonctionnelles et des dépendances d'inclusion. Ces dépendances sont identifiées en utilisant des techniques de fouille de données. Les dépendances seront principalement utilisées pour identifier les dimensions. Les données sont ensuite classées en trois catégories : les mesures, les clés et les données descriptives. Un algorithme est proposé pour générer un schéma en flocon de neige en analysant les métadonnées des bases : (1) les tables de faits sont identifiées par un processus semi-automatique, selon les cardinalités des relations et le nombre de mesures identifiées. L'utilisateur valide ensuite le résultat. (2) Les dépendances d'inclusion identifiées sont représentées par différents graphes connexes. Le graphe est considéré comme une dimension si une dépendance existe entre une table de faits et un nœud du graphe. Ce nœud sera considéré comme le premier niveau de la hiérarchie de dimension. Les hiérarchies de dimensions sont analysées afin de vérifier l'agrégation des données à travers chaque hiérarchie.

Giorgini et al. [65] proposent de définir le schéma conceptuel multidimensionnel de l'ED à partir de l'analyse des besoins qui s'effectue selon une approche orientée agent basé sur le Framework i*. Ce framework repose sur les notions d'*agent* et de *but* dans les différentes phases de développement d'un système. Le Framework permet d'identifier les utilisateurs du système et de les modéliser en acteurs sociaux dépendants les uns des autres à travers les buts à réaliser, les tâches à effectuer et les ressources à utiliser. Les auteurs précisent que cette approche peut s'arrêter à ce niveau si l'utilisateur ne souhaite pas considérer les sources de données. Dans le cas d'une approche mixte ou hybride, les concepts multidimensionnels du schéma conceptuel sont mis en correspondance avec les entités des schémas des sources. Deux types de schémas sont étudiés : les schémas relationnels et les schémas E/A. Cette mise en correspondance est effectuée manuellement par le concepteur en associant chaque fait, dimension ou mesure du schéma conceptuel à une table ou à un attribut dans les sources. Les hiérarchies de dimensions sont construites en identifiant les dépendances fonctionnelles entre les différents attributs ou relations. Le schéma est ensuite présenté aux utilisateurs pour une deuxième validation.

Prat et al. proposent dans [154] une méthode pour dériver le schéma conceptuel, logique

et physique d'un ED selon les trois niveaux d'abstraction recommandés par l'architecture ANSI /SPARC. À chaque étape, les auteurs introduisent un méta-modèle et un ensemble de règles de transformation définies en langage OCL (Object Contraint Language), permettant le mapping entre les méta-modèles. Les trois phases de modélisation sont conduites comme suit : (1) la phase de modélisation conceptuelle commence par collecter les besoins des utilisateurs selon les techniques classiques utilisées pour les BD comme les interviews, le prototypage, etc. Le modèle conceptuel multidimensionnel de l'ED est ensuite défini sous forme d'un diagramme de classes. (2) La phase de modélisation logique consiste à générer le schéma logique de l'ED automatiquement à partir du modèle conceptuel enrichi, en appliquant un ensemble de règles de transformation. (3) La phase de modélisation physique consiste à traduire le schéma logique multidimensionnel en un schéma de base de données physique en fonction de l'outil de mise en œuvre cible. Les auteurs ont opté pour le SGBD Oracle MOLAP. Les auteurs présentent une méthode similaire dans [153] supportant une modélisation physique ROLAP.

Mazon et al. [134] présentent une approche semi-automatique hybride pour définir le schéma conceptuel de l'ED. Les besoins sont exprimés selon le Framework i*. Le schéma conceptuel multidimensionnel obtenu à partir des besoins est représenté par un modèle représentant une extension du modèle UML. Ce schéma est ensuite validé par les sources de données en utilisant des relations Query/View/Transformation (QVT). QVT est un langage standardisé pour exprimer des transformations de modèles. Les sources utilisées sont des sources relationnelles. Les règles QVT sont utilisées pour valider, de manière semi-automatique, le modèle conceptuel multidimensionnel obtenu à partir des besoins, en l'alignant avec les schémas des sources. Cette validation est basée sur les formes normales multidimensionnelles définies par *Husemann et al.* [88]. Cinq principales règles QVT sont définies pour permettre d'aligner le schéma conceptuel multidimensionnel avec les schémas relationnels des sources de données : (1) chaque dépendance fonctionnelle dans le schéma conceptuel doit correspondre à une dépendance fonctionnelle dans les schémas relationnels. (2) Les dépendances fonctionnelles entre les niveaux de dimension identifiées dans les sources de données doivent être représentées par des relations d'agrégation dans le schéma conceptuel. (3) les mesures qui peuvent être dérivées à partir d'autres mesures doivent être identifiées dans le schéma conceptuel. (4) Des mesures doivent être affectées à des faits. Les mesures représentant les clés du fait doivent être identifiées. (5) les valeurs nulles sont identifiées dans les sources et traitées dans le schéma conceptuel.

Song et al. [184] présentent une méthode automatique orientée sources permettant de définir le schéma logique de l'ED à partir des schémas E/A des sources de données. Le processus de conception suit les étapes suivantes : (1) les schémas E/A des sources sont redéfinis afin de transformer les associations ternaires en des associations binaires. (2) Les entités ayant le nombre d'associations 'un à plusieurs' supérieur à une valeur seuil, sont considérées comme des faits. (3) Les entités liées aux entités faits par des relations un à plusieurs sont considérées comme les dimensions du fait. Les auteurs utilisent l'ontologie terminologique Wordnet afin d'identifier les hiérarchies de dimensions.

Romero et al. [162, 164] présentent une méthode semi-automatique de conception multidimensionnelle d'un ED, basée sur une ontologie décrite en langage OWL (Ontology Web Language). L'ontologie représente le domaine d'intérêt de l'ED. Elle est supposée intégrant un ensemble de sources. Cette méthode étudie les multiplicités entre les concepts de l'ontologie pour

définir les concepts multidimensionnels faits et dimensions du schéma de l'entrepôt. Un concept de l'ontologie est considéré comme un fait potentiel s'il est lié à un nombre important de dimensions et de mesures. Les mesures sont désignées comme les attributs numériques permettant l'agrégation des données. Ils sont liés au concept représentant le fait par une relation un à un. Un concept de l'ontologie est considéré comme dimension potentielle s'il est rattaché à un fait par une relation un à plusieurs. Les relations un à plusieurs sont représentées dans les ontologies OWL par des relations (OWL :ObjetcProperty) de type fonctionnel (Functional ObjetcProperty). Les hiérarchies de dimensions sont déterminées en recherchant les relations un à plusieurs entre les concepts identifiés comme dimensions. Cette identification peut se faire de manière automatique. Les résultats sont ensuite présentés au concepteur qui doit les valider. Le schéma en flocon de neige est ensuite défini.

Nebot et al. proposent dans [145] de fournir un Framework permettant d'analyser les données fournies par le web sémantique sous forme d'annotations (méta données xml ou Rdf). Ils proposent ainsi de concevoir un ED semi-structuré, qui stocke les ressources du web, les ontologies de domaines et les annotations sémantiques utilisant ces ontologies. Le concepteur commence par identifier manuellement dans les ontologies de domaine, les concepts multidimensionnels constituant des faits, dimensions, mesures et les relations roll up entre les hiérarchies de dimensions. Une fois ces concepts choisis, des ontologies applicatives constituant ces concepts sont générées automatiquement. Les liens entre les ontologies applicatives et les ontologies de domaines sont sauvegardés sous forme d'axiomes. La génération des ontologies applicatives s'effectue en utilisant le langage OntoPath (proposé par les mêmes auteurs) qui est un langage permettant de rechercher et de spécifier un fragment d'ontologie. Le schéma de l'ontologie applicative obtenu est validé en vérifiant un ensemble de contraintes multidimensionnelles (comme l'additivité, l'orthogonalité, etc). La dernière étape consiste à charger les instances ontologiques dans l'entrepôt de données, et de générer les cubes multidimensionnels.

2.2.2 Analyse des travaux de conception du schéma de l'ED

Après une revue de la littérature des principaux travaux de conception du schéma d'ED, nous procédons à l'analyse de ces travaux. Nous avons pour cela établi un certain nombre de critères *pour chacune des phases de conception*, comme illustré dans la figure 2.1. Un tableau récapitulatif synthétise l'analyse des travaux selon les critères retenus. Nous concluons cette section par l'étude du positionnement de notre contribution par rapport aux travaux cités.

2.2.2.1 Définition des besoins

Nous considérons pour ce premier critère, la définition des besoins selon les deux méthodes utilisées pour collecter les besoins : la collecte orientée utilisateur et la collecte orientée sources.

Pour le premier type de collecte *orientée besoins des utilisateurs*, nous proposons de classer les travaux retenus selon trois principaux critères : le *type de besoins considérés*, la *représentation des besoins* et le *niveau d'abstraction de la spécification des besoins proposée.*

2.2.2.1.1 Types de besoins

L'analyse des besoins dans les méthodes de conception d'ED diffère selon l'objet analysé. On

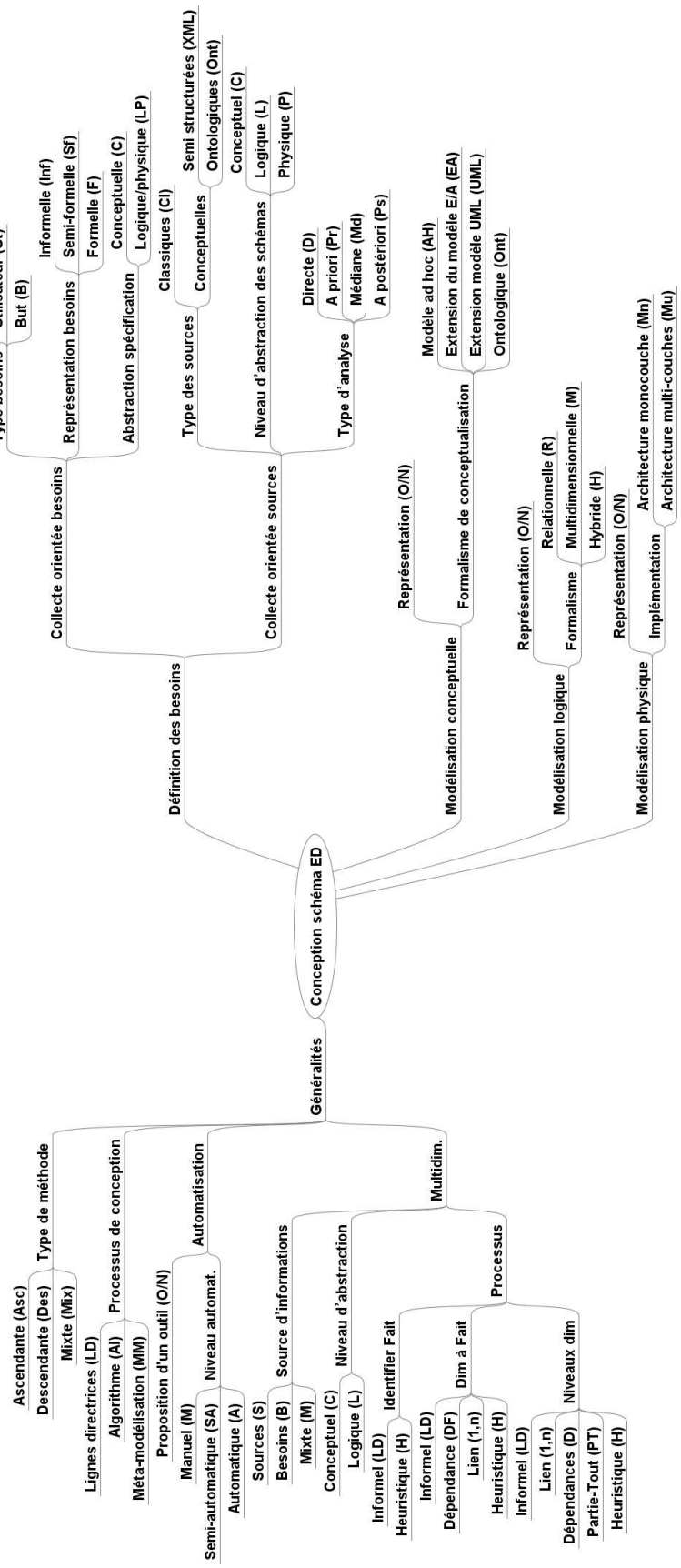

FIGURE 2.1 – Critère de classification des travaux de conception du schéma de l'ED

retrouve une analyse orientée processus, orientée utilisateur ou orientée but. L'analyse orientée *processus* [122, 169, 29, 109] analyse les besoins en identifiant les processus métiers de l'organisation. Un processus métier est décrit comme un ensemble d'activités ordonnées dans l'espace et le temps (s'effectuant dans un intervalle précis de temps) ayant des entrées et des sorties bien définies. L'analyse orientée *utilisateurs* [213, 32] s'intéresse à identifier les utilisateurs cibles et à spécifier leurs besoins individuels pour les intégrer dans un modèle de besoins unifié. L'analyse orientée *but* [25, 65, 62, 152, 133, 105] identifie les buts et objectifs des utilisateurs qui guident les décisions à différents niveaux de l'organisation. Plusieurs méthodes de conception d'ED utilisent une approche orientée but pour la définition des besoins des utilisateurs. Plusieurs auteurs reconnaissent que cette approche offre une meilleure définition des besoins des utilisateurs [25, 160]. De plus, l'analyse orientée but permet de valider les besoins collectés en identifiant les buts conflictuels. Elle permet de montrer les différentes alternatives de modélisation pour atteindre un but donné [64]. Identifier les objectifs et buts des utilisateurs en début de projet est une étape cruciale dans le processus de développement, spécialement dans les projets d'applications décisionnelles qui visent à analyser l'activité d'une organisation et où la réalisation des buts est un indicateur important de cette activité [185].

2.2.2.1.2 Représentation des besoins

Nous rappelons que le processus d'ingénierie des besoins suit les quatre *phases* suivantes : (1) la phase de collecte des besoins auprès des utilisateurs du système final, (2) la phase de spécification des besoins, (3) la phase de validation des besoins et (4) la phase de gestion des besoins. L'étape de *spécification* des besoins présente une étape essentielle permettant de fournir le modèle de besoins. Ce dernier peut être défini selon différentes *représentations* : en utilisant des modèles *informels*, *semi-formels* ou *formels*.

Les représentations *informelles* comme le modèle MAP proposé par [62], sont construites en langue naturelle avec ou sans règles de structuration. Leur utilisation introduit des risques d'ambiguïté car ni leur syntaxe, ni leur sémantique ne sont parfaitement définies.

Les modèles *semi-formels* sont généralement basés sur des notations graphiques avec une syntaxe précise et permettent d'avoir une vision claire du système. Ces modèles présentent de bons vecteurs de communication entre les concepteurs et les utilisateurs du système. Par exemple, certains travaux représentent les besoins des utilisateurs d'un ED par les modèles de la notation UML [122, 29]. D'autres travaux [65, 133] représentent les besoins des utilisateurs par les modèles fournis par le Framework i*. D'autres travaux proposent des modèles ad hoc [70, 88]

Les modèles formels sont basés sur des notations mathématiques qui fournissent un cadre précis et non ambigu pour la modélisation des besoins. Par exemple, certains travaux proposent de spécifier les besoins des utilisateurs en utilisation des langages formels comme SQL ou MDX [163, 150]. D'autres travaux comme [106, 104, 105] proposent de spécifier besoins des utilisateurs en utilisant une ontologie OWL, qui permet de fournir une représentation formelle des besoins collectés. Nebot et al. [144] suit cette représentation en proposant d'exprimer les besoins des utilisateurs par des expressions formalisées en logiques de description.

2.2.2.1.3 Niveau d'abstraction du modèle de besoins

Ce critère indique le niveau d'abstraction (conceptuel, logique ou physique) du modèle de

besoins proposé. La plupart des travaux proposent de spécifier les besoins des utilisateurs au niveau conceptuel [70, 154, 104], généralement par des modèles UML. D'autres travaux spécifient les besoins au niveau logique ou physique en utilisant le langage SQL [163] ou MDX [150].

Pour le deuxième type de collecte *orientée sources*, nous proposons de classer les travaux retenus selon trois principaux critères suivants : le *type des sources*, le *niveau d'abstraction des schémas des sources* et le *type d'analyse des sources*.

2.2.2.1.4 Type de sources

Ce critère indique le type de sources considérées par la méthode de conception qui peuvent être : classiques ou ontologiques. La plupart des travaux considèrent des sources de données *classiques* généralement des BD relationnelles. Quelques travaux considèrent des sources *sémantiques* représentées par des schémas XML [209], des annotations du web sémantique [145] ou des BD à base ontologique [105, 23].

2.2.2.1.5 Le niveau d'abstraction des schémas des sources

Ce critère considère le niveau d'abstraction conceptuel, logique ou physique des schémas des sources considérées. Quelques travaux considèrent des sources de données représentées par leur schéma *logique* (généralement par leur schéma relationnel) [92, 184, 134, 10]. Le modèle logique de données dépend de contraintes d'implémentation qui peuvent altérer la qualité du schéma de l'ED généré. Afin de remédier à cet inconvénient, d'autres méthodes proposent de définir le schéma de l'ED à partir des schémas *conceptuels* des sources de données (généralement des schémas E/A) comme [138, 184, 24, 209]. Des méthodes plus récentes suggèrent l'utilisation des schémas *ontologiques* des sources de données [162, 164, 145, 23, 18].

2.2.2.1.6 Le type d'analyse des sources

Ce critère distingue quatre types d'analyse des sources : une analyse directe, une analyse à priori, une analyse médiane et une analyse à postériori.

Les travaux proposant une analyse *directe* sont les méthodes de conception exclusivement orientées sources [88, 92, 184].

Les autres approches sont les approches proposant une confrontation ou une mise en correspondance entre les données des sources et les besoins des utilisateurs. Les sources de données restent cependant au cœur de la construction de l'ED. Un mapping entre les sources et les besoins doit être effectué, et peut se faire de trois manières : à priori, à postériori, ou selon une approche médiane.

Dans une approche *à postériori* (comme dans [25]), un schéma multidimensionnel de l'ED est défini à partir des sources de données. Un autre schéma multidimensionnel est généré à partir de l'analyse des besoins. Un mapping entre ces deux schémas est ensuite effectué afin de générer le schéma final de l'ED (figure 2.2). Ce type de méthodes présente deux principaux inconvénients : (1) la possibilité de générer un nombre de schémas inutiles qui ne seront pas exploités. (2) La difficulté d'effectuer un mapping entre les schémas qui est due aux divers problèmes d'hétérogénéité entre ces schémas.

Dans une approche *médiane*, un seul schéma conceptuel est généré à partir des sources (resp. besoins) et est validé par les besoins (resp. sources). La figure 2.3 illustre cette approche. Dans

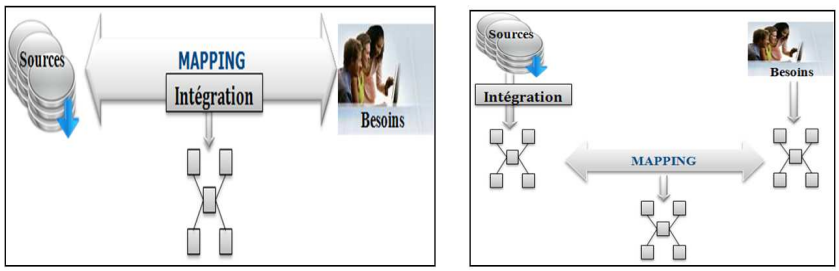

FIGURE 2.2 – Confrontation sources et besoins selon une approche à priori et à postériori

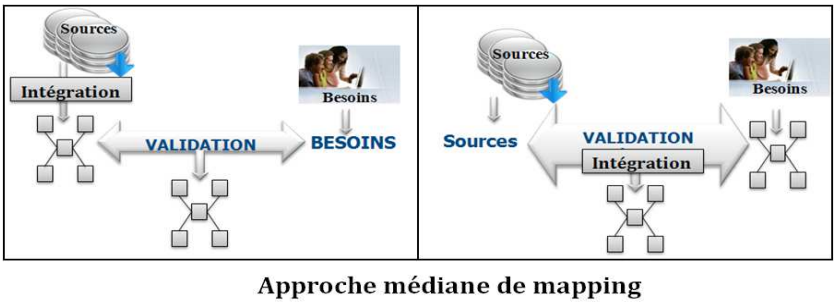

FIGURE 2.3 – Confrontation sources et besoins selon une approche médiane

les deux cas, un inconvénient majeur réside dans l'exploitation minime d'une des sources d'informations (sources de données ou besoins utilisateurs). Dans le premier cas (comme dans [150]), un schéma initial de l'ED est défini à partir de l'analyse des sources, qui est ensuite validé par les besoins des utilisateurs. Dans le deuxième cas (comme dans [65]), un schéma initial de l'ED est défini à partir des besoins. Il est ensuite validé en analysant les sources. Cette validation est généralement effectuée manuellement par le concepteur avec la possibilité d'impliquer les utilisateurs.

Dans une approche *à priori*, un mapping entre les sources et les besoins est effectué en début de conception avant la génération du schéma de l'ED [163, 163, 104, 105] (figure 2.2). Ce type de mapping permet d'éliminer les inconvénients cités pour les approches d'analyse précédentes.

2.2.2.2 Modélisation conceptuelle

Pour la deuxième phase de modélisation conceptuelle, nous proposons de distinguer les critères de classification suivants : la représentation effective d'une phase de modélisation conceptuelle (O pour Oui et N pour Non) et le formalisme de conceptualisation.

Du point de vue de leur *formalisme de conceptulisation*, les méthodes existantes peuvent être classées en quatre catégories : les modèles *ad hoc* [70, 88], les formalismes adoptant ou étendant *le modèle E/A* [60, 168, 194], les formalismes adoptant ou étendant *les modèles UML* [5, 126, 134] et les modèles *ontologiques* [104, 145]. Les formalismes de conceptualisation proposés permettent, pour la plupart, de représenter tous les concepts de base du modèle multidimensionnel de l'ED (fait, dimension, mesures, attributs de dimensions, hiérarchie de dimension) et ont le même noyau d'expressivité [158]. Ces formalismes diffèrent cependant dans la possibilité de représenter des

concepts multidimensionnels plus avancés tels que les hiérarchies irrégulières ou l'additivité des mesures.

2.2.2.3 Modélisation logique

Nous retenons pour cette phase de modélisation deux principaux critères : la représentation effective d'une phase de modélisation logique (O pour Oui et N pour Non) et le type de schéma logique fourni en sortie de cette phase. Deux types de schémas sont généralement choisis : *relationnel* ou *multidimensionnel* (sous forme d'un tableau ou d'un cube). La plupart des méthodes proposées optent pour une modélisation relationnelle. Dans les implémentations relationnelles, les schémas en étoile [164, 126] et en flocon de neige [164] sont largement utilisés.

Les méthodes proposant des représentations logiques multidimensionnelles sont moins nombreuses. Quelques structures de données multidimensionnelles ont été proposées où l'agrégation multidimensionnelle est formalisée en langage SQL en introduisant un nouvel opérateur CUBE pour le cubes multidimensionnels. Cet opérateur peut être vu comme l'équivalent du Group by pour les tables relationnelles [7]. Des travaux se sont intéressés à l'étude des relations entre les cuboïdes pour optimiser les requêtes posées sur le cube. Des structures condensées ont ainsi été proposées telles que des cubes condensés [59], les dwarfs [179], et CQ-Tree [115] pour représenter et gérer les cubes de données.

2.2.2.4 Modélisation physique

Pour cette dernière phase de conception, nous retenons comme critères : la représentation effective d'une phase de modélisation physique et le type de déploiement de l'ED. Nous nous intéressons aux deux types de déploiement suivants : déploiement dans un *ED classique monocouche* (représentant le modèle physique de données uniquement) ou déploiement selon un *ED multicouche* faisant persister le modèle physique de données et également une partie de la sémantique du modèle.

La plupart des travaux implémentent et déploient leur ED via des SGBD relationnels en utilisant des architectures monocouches classiques par opposition aux architectures ontologiques. Nous appelons les architectures multicouches, celles permettant de stocker l'ontologie définissant la sémantique (ou une partie de la sémantique) des données stockées. Deux principaux travaux proposent des méthodes de conception implémentant l'ED dans une architecture sémantique ([104] et [145]).

2.2.2.5 Critères généraux

D'autres critères représentant des aspects généraux de la conception sont retenus dans notre classification. Ces critères sont les suivants :

Le type de la méthode proposée : les trois types de méthodes peuvent être utilisés : méthode orientée *sources* (ou ascendante) [88, 92, 184], méthode orientée *besoins* (ou descendante) [107, 154], méthode *hybride* ou *mixte* [138, 70].

Le processus de conception : certains travaux définissent des *lignes directrices* générales pour concevoir l'ED [107, 33, 138]. D'autres travaux formalisent le processus de conception par

un ou plusieurs *algorithmes* [70, 150]. Quelques autres travaux proposent une approche de *méta-modélisation* pour concevoir l'ED [154, 134]. Le but d'une approche de méta-modélisation est de fournir une approche automatisée du processus de conception et dans certains cas d'assurer une traçabilité des besoins de l'ED [130, 131].

L'automatisation du processus de conception : nous distinguons deux critères d'automatisation : la proposition d'un *outil* support à la méthode de conception proposée [92, 184, 97], et le *niveau d'automatisation* du processus de conception. Le processus de conception défini peut être complètement *automatique* [154, 184], *semi-automatique* [25, 209, 105] ou *manuel* nécessitant la présence d'un expert ou d'un concepteur [213, 33].

L'annotation multidimensionnelle : la définition du processus de conception des ED est structurée autour d'une modélisation multidimensionnelle. Trois principaux critères sont retenus pour classifier les travaux proposés.

Le premier critère concerne la *source d'informations* à partir de laquelle se fait l'annotation multidimensionnelle : à partir des sources de données [184, 164, 145] ou à partir des besoins des utilisateurs [88, 154], ou à partir des deux ces sources d'informations [65, 105].

Le deuxième critère concerne le *niveau d'abstraction* du processus d'annotation multidimensionnelle, i.e au niveau de la phase de modélisation conceptuelle [70, 65, 105] ou logique [107, 33, 92, 164].

Le troisième critère concerne le *processus* d'annotation multidimensionnelle. Après analyse des différentes méthodes de conception, nous retenons un certain nombre d'étapes communes suivies par ces méthodes pour l'annotation multidimensionnelle consistant à : identifier les faits et leurs mesures, identifier les dimensions et leur interaction avec les faits, et enfin à identifier les relations entre les niveaux d'agrégation dans une hiérarchie de dimension [132]. Nous analysons ci-après les méthodes proposées selon ces étapes. Certains travaux n'étudient pas la représentation des hiérarchies de dimensions. La plupart des travaux commencent par identifier les faits et leur associent des dimensions. Quelques autres travaux identifient d'abord les dimensions [109]. Quelques approches étudient les structures multidimensionnelles complexes comme les hiérarchies multiples ou généralisées [129]. Nous nous intéressons aux situations régulières uniquement.

- *Identification des faits* : cette étape peut se faire de diverses manières : de manière *informelle* sous forme de lignes directrices, en utilisant des *heuristiques* pour déterminer les concepts candidats à représenter des faits [150, 92, 184]. Par exemple, *Phipps et al.* [150] considèrent les entités contenant des attributs numériques comme des candidates potentielles. *Song et al.* [184] considèrent les entités ayant un nombre élevé de relations un à plusieurs entrantes comme des candidates potentielles.
- *Représenter les faits et leurs dimensions associées* : la majorité des méthodes de conception présentent le lien entre un fait et sa dimension par l'existence d'une dépendance fonctionnelle (DF) lorsque le schéma est relationnel, i.e. une instance d'un fait détermine fonctionnellement une instance d'une dimension, ou par l'existence d'une relation un à plusieurs entre le fait et sa dimension (lorsque le schéma est conceptuel ou ontologique).
- *Expliciter les hiérarchies de dimensions* : on retrouve différentes sémantiques des relations entre les niveaux d'agrégation dans une hiérarchie de dimension. La plupart des travaux [118, 138, 162, 164] définissent ces hiérarchies de dimensions en détectant les relations un à plusieurs entre les entités ou tables identifiées comme dimensions au niveau des sources.

D'autres travaux [33] identifient les relations entre niveaux de dimensions comme des fonctions d'ordre partiel. *Levene et al.* [121] identifient les relations entre les niveaux d'une dimension comme des dépendances d'inclusion hiérarchiques entre un niveau et son niveau inférieur. *Niemi et al.* [147] définissent d'autre types de dépendances (fonctionnelles, d'inclusion et booléennes) entre les niveaux de dimensions. *Hacid et al.* [80] proposent une autre sémantique des relations entre les niveaux de dimensions en les représentant par la relation part-whole (Partie-Tout) où un niveau d'une dimension fait partie d'un niveau supérieur de la même dimension.

Mazon et al. [132] proposent une étude exhaustive des différentes représentations multidimensionnelles où les auteurs font ressortir les situations régulières fournissant un modèle multidimensionnel correctement agrégeable.

Selon les auteurs, la représentation adéquate des interactions entre les dimensions et les faits revient à déterminer le grain adéquat des faits. Un modèle multidimensionnel doit avoir une granularité uniforme, ce qui signifie que toute mesure dans le fait doit être déterminée par toutes les dimensions [109]. Les situations régulières des relations liant un fait à sa dimension consistent à utiliser les multiplicités (0,n) du coté du fait, et (1,1) du coté de la dimension.

Selon les mêmes auteurs, la représentation adéquate des relations entre les niveaux d'agrégation dans une hiérarchie de dimension permet d'assurer des agrégations correctes des données. Ce principe est représenté par la notion de "summarizability", qui signifie la situation dans laquelle le résultat d'une agrégation peut être calculé en utilisant les agrégations précédentes [116]. Par exemple, la mesure "nombre d'articles vendus par mois" peut être agrégée, par contre la mesure "nombre d'articles restant en stock par mois" ne peut pas être agrégée à cause des valeurs redondantes du stock. Cette notion a été introduite par *Rafanelli et Shoshani* [156] dans le contexte des BD statistiques. Les auteurs fournissent les bases pour identifier et éviter les problèmes d'agrégation dans un espace multidimensionnel. Se basant sur cette étude, *Lenz et Shoshani* [119] définissent trois principales conditions nécessaires (intuitivement suffisantes aussi) assurant une agrégation correcte [132] :

1. Disjonction : indique que les associations 'plusieurs à plusieurs' entre les niveaux de dimensions ne doivent pas être utilisées.
2. Complétude : indique qu'il ne doit pas y avoir des valeurs manquantes dans les dimensions. Les auteurs rajoutent la condition nécessitant que la multiplicité minimale des deux extrémités des associations liant les niveaux de dimension doit être égale à 1 (et non à 0).
3. Compatibilité : il doit y avoir une compatibilité sémantique entre le niveau de dimension, l'attribut mesure et la fonction d'agrégation appliquée.

Pour récapituler, les situations régulières des relations entre les dimensions assurant une agrégation correcte des données consistent à utiliser les multiplicités (1,n) du coté de la dimension de niveau inférieur, et (1,1) du coté de la dimension de niveau supérieur. Pour le cas particulier où la fonction d'agrégation est limitée à l'opérateur Somme, le concept "d'additivité" est utilisé au lieu "summarizability". Une mesure est dite additive pour une dimension s'il est possible d'utiliser l'opération d'addition Somme pour agréger cette mesure tout au long des hiérarchies de cette dimension.

Différents travaux considèrent les trois conditions énoncées par *Lenz et Shoshani* comme une première étape permettant d'assurer la qualité d'un schéma d'ED. Certains travaux comme

[88, 116, 118, 117] se sont basés sur ces conditions afin de définir des formes normales multidimensionnelles, à l'instar des formes normales définies pour les BD relationnelles.

Les tableaux 2.1 et 2.2 synthétisent les principaux travaux cités et les analysent selon ces critères de comparaison. Dans ces tableaux, la mention "NM" signifie "critère défini mais non mentionné", et la mention "-" signifie "critère non défini". Nous étudions ci-après le positionnement de notre première proposition par rapport à l'ensemble de ces travaux selon les critères retenus.

Travaux/Critères	[107]	[33]	[70]	[24]	[88]	[138]	[25]	[150]	[209]
Définition des besoins									
Collecte orientée besoins									
Type de besoins	Pr	NM	NM	B	-	NM	B	Ut	-
Représentation besoins	Inf	Inf	Inf	Inf	-	NM	Sf	F	-
Abstraction Spécification	C	C	C	C	-	NM	C	LP	-
Collecte orientée sources									
Type de sources	Cl	Cl	Cl	Cl	Cl	Cl	Cl	Cl	XML
Niveau abstraction schémas	L	C	C	C	C	C	C	R	C
Type analyse	Pr	Pr	Pr	Pr	D	Pr	Ps	Md	D
Modélisation conceptuelle									
Représentation	N	N	O	N	O	O	O	O	N
Formalisme	-	-	AH	-	EA	EA	AH	EA	-
Modélisation logique									
Représentation	O	O	O	O	O	O	O	N	O
Formalisme	R	R,M	R	R	R	R	R	-	R
Modélisation physique									
Représentation	O	N	O	N	N	N	N	N	N
Implémentation	-	Mn	Mn	-	-	-	-	-	-
Critères généraux									
Type méthode	Des	Mix	Mix	Mix	Asc	Mix	Mix	Mix	Mix
Processus de conception	LD	LD	Al	LD	LD	LD	Al	Al	Al
Automatisation									
Niveau automatisation	M	M	SA	SA	M	M	SA	SA	SA
Proposition outil	N	N	O	O	N	N	N	N	O
Annotations multidimensionnelles									
Source d'annotation	B	B/S	S	S/B	B	S/B	S/B	S	S
Niveau abstraction	L	L	C	L	C	C	C	C	L
Processus d'annotation									
Identification du fait	LD	LD	H	H	LD	LD	H	H	H
Lien Fait et Dimensions	LD	LD	DF	DF	LD	(1,n)	(1,n)	H	FD
Hiérarchies de dimensions	LD	H	(1,n)	D	D	(1,n)	-	(1,n)	(1,n)

TABLE 2.1 – Synthèse de comparaison entre les travaux de conception du schéma de l'ED

2.2.3 Positionnement de nos contributions

L'analyse des différents travaux cités dans la littérature relative à la conception du schéma de l'ED nous permet de positionner nos contributions selon les critères retenus.

De part les *aspects généraux*, nous positionnons notre méthode comme une méthode *mixte* et

Travaux/Critères	[213]	[92]	[65]	[154]	[134]	[184]	[162, 164]	[145]	[106, 105]
Définition des besoins									
Collecte orientée besoins									
Type de besoins	NM	-	B	Ut	B	-	NM	-	B
Représentation besoins	Sf	-	Sf	Sf	Sf	-	NM	-	F
Abstraction Spécification	C	-	C	C	C	-	NM	-	C
Collecte orientée sources									
Type de sources	NM	Cl	Cl	-	Cl	Cl	Cl	Ont	Ont
Niveau abstraction schémas	C	L	C,L	-	L	C	C	C	C
Type analyse	Pr	D	Md	-	Md	D	D	D	Pr
Modélisation conceptuelle									
Représentation	O	N	O	O	O	N	N	O	O
Formalisme	NM	-	AH	UML	AH	-	-	Ont	Ont
Modélisation logique									
Représentation	N	O	N	O	N	O	O	N	O
Formalisme	-	R	-	R,M	-	R	R	-	R
Modélisation physique									
Représentation	N	N	N	O	N	N	N	O	O
Implémentation	-	-	-	Mn	-	-	-	Mu	Mu
Critères généraux									
Type méthode	Mix	Asc	Mix	Des	Mix	Asc	Mix	Asc	Mix
Processus de conception	LD	Al	Al	MM	MM	Al	Al	Al	Al
Automatisation									
Niveau automatisation	M	SA	SA	A	SA	A	SA	SA	SA
Proposition outil	N	O	N	O	O	O	N	N	O
Annotations multidimensionnelles									
Source d'annotation	B	S	B,S	B	B	S	S	S	B,S
Niveau abstraction	NM	L	C	C	C	L	L	C	C
Processus d'annotation									
Identification du fait	NM	H	H	H	LD	H	H	NM	H
Lien Fait et Dimensions	NM	DF	DF	H	FD	(1,n)	DF	NM	(1,n)
Hiérarchies de dimensions	NM	D	D	NM	(1,n)	H	(1,n)	NM	(1,n)

TABLE 2.2 – Synthèse de comparaison entre les travaux de conception du schéma de l'ED

semi-automatique où le concepteur et/ou utilisateurs interviennent à des étapes clés lorsque l'expertise du concepteur et la validation des utilisateurs sont nécessaires. Le processus de conception est *formalisé* par plusieurs algorithmes. Nous estimons que le processus d'annotation multidimensionnelle doit être effectué dès la phase *conceptuelle*, en s'appuyant sur les deux modèles (sources et besoins) qui sont définis indépendamment de toute contrainte d'implémentation. Un algorithme d'annotation multidimensionnelle est fourni dans la méthode.

Nous positionnons ensuite notre méthode par rapport aux phases du cycle de conception des ED. Dans la première phase de "définition des besoins de l'ED", plusieurs points sont remarqués. La plupart des travaux qui optent pour une collecte orientée besoins des utilisateurs utilisent généralement l'approche orientée but qui s'adapte parfaitement au contexte des ED et projets décisionnels. Les décideurs et utilisateurs finaux des applications décisionnelles expriment naturellement leurs besoins sous forme de buts et d'objectifs à atteindre. Nous optons donc pour

une approche *orientée but*. D'un point de vue représentation des besoins des utilisateurs, nous remarquons un manque de représentation formelle du modèle des besoins dans plusieurs travaux. La validation des besoins est, par conséquent, effectuée manuellement par le concepteur et/ou les utilisateurs du système. Une représentation formelle du modèle de besoins permettrait de fournir une validation de ce modèle. Nous proposons dans nos contributions de représenter le modèle de besoins *au niveau ontologique*, et nous montrons que cette représentation permet de raisonner sur le modèle de besoins afin de le valider et d'inférer de nouvelles connaissances. L'utilisation des ontologies permet de plus d'offrir une représentation conceptuelle des besoins indépendante de toute contrainte d'implémentation.

Lorsque la collecte des besoins de l'ED est orientée sources de données, la plupart des travaux utilisent des sources de données classiques monocouches généralement relationnelles. Certains travaux génèrent le schéma conceptuel de ces sources pour avoir une vue plus abstraite du schéma de données. Aussi, l'analyse des sources est généralement confrontée aux besoins des utilisateurs dans les derniers travaux de conception des ED proposés. Nous proposons une méthode de conception *mixte* effectuant une confrontation (ou mise en correspondance) entre les données des sources et les besoins des utilisateurs *à priori* (dès le début de la conception). Les deux types de sources de données (classiques monocouches et multicouches) seront considérés. Les schémas des sources sont définis au niveau *conceptuel*.

La phase qui suit la définition des besoins est la phase de "modélisation conceptuelle". Cette phase n'a pas toujours été considérée dans les travaux de conception des ED. Elle est actuellement reconnue comme une phase faisant partie du cycle de conception de l'ED. Le schéma conceptuel de l'ED est défini selon un formalisme de conceptualisation. Nous introduisons dans notre méthode la phase de modélisation conceptuelle. Nous proposons d'exploiter un formalisme *ontologique* qui possède un pouvoir expressif plus important que les formalismes conceptuels proposés (E/A, UML).

La phase de "modélisation logique" est souvent effectuée de manière automatique par la définition d'un algorithme et de règles de passage du schéma conceptuel au schéma logique. Ce dernier suit généralement une représentation relationnelle. Nous proposons une approche *générique* qui permet un déploiement *à la carte* du schéma de l'ED selon la représentation choisie par l'utilisateur.

Pour la dernière phase de "modélisation physique", nous remarquons que les travaux étudiant cette phase sont peu nombreux et proposent habituellement de représenter le schéma de l'ED selon une architecture classique monocouche. Nous montrons dans notre approche que le schéma de l'ED peut être représenté au niveau physique dans une architecture *classique monocouche* et *ontologique multicouche*. Cette dernière permet de stocker également le schéma logique et le schéma conceptuel de l'ED. Nous fournissons de plus un mécanisme de persistance des besoins des utilisateurs dans cette architecture. La dernière colonne du tableau 2.2 récapitule le positionnement de notre première proposition par rapport aux travaux cités.

2.3 Conception ETL de l'ED

2.3.1 Etat de l'art des travaux de conception

Nous présentons dans cette section les principaux travaux portant sur l'étude de la phase de conception ETL des ED.

Kimball et al. [108] décrivent les principales activités devant être effectuées pour l'extraction des données des sources, leur nettoyage et transformation et leur chargement dans l'ED. Les schémas des sources, de l'ED et les mappings sont définis au niveau physique. Les auteurs ne fournissent pas une méthode détaillée pour la conception, l'alimentation et le déploiement de l'ED. L'approche proposée peut être considérée comme une documentation informelle de l'ensemble des activités de conception du schéma de l'ED et des activités ETL.

Calvanese et al. proposent dans [36, 37] une approche d'intégration et de réconciliation des données dans un ED, où les schémas des sources de données et le schéma cible de l'ED sont alignés à un schéma intermédiaire défini au niveau conceptuel. L'approche est caractérisée par les aspects suivants : (1) la définition d'un schéma global intermédiaire au niveau conceptuel en utilisant le formalisme E/A enrichi par le langage DLR de la famille des logiques de description. (2) Les schémas locaux (sources de données et entrepôt cible) sont exprimés au niveau logique via le modèle relationnel. (3) Des correspondances sont définies entre les schémas locaux (des sources de données et de l'ED) et le schéma conceptuel intermédiaire. Trois types de correspondances sont cités : les *conversions* permettant la résolution des conflits, les *matchings* permettant la correspondance entre les données, et les *réconciliations* permettant la vérification de la cohérence des données avant leur chargement dans l'ED cible. L'approche est basée sur un algorithme de réécriture des requêtes dont le rôle est de reformuler les requêtes de l'ED sur les schémas des sources.

Stohr et al. proposent dans [186] un méta-modèle d'intégration pour la gestion des métadonnées dans un ED. Le modèle est représenté par un diagramme de classe UML et représente l'ensemble des informations sur la structure, le contenu et les interdépendances des composants d'un ED. Les deux niveaux de métadonnées : *techniques* (décrivant le modèle physique des sources et de l'ED) et *sémantiques* (décrivant les vues multidimensionnelles des cubes) sont représentés dans le méta-modèle. La partie de mappage est représentée par deux classes : Mapping et Transformation qui permettent de décrire les flux de transfert des données des sources opérationnelles à l'entrepôt et aux cubes OLAP. Les deux classes sont également employées pour représenter les dépendances entre les concepts multidimensionnels et leur représentation physique (données) dans l'ED. Ces mapping sont principalement utilisés pour faciliter l'interprétation des requêtes multidimensionnelles en requêtes SQL exécutables sur les tables de l'ED.

Labio et al. proposent dans [114] un Framework permettant de décrire les activités de chargement de données (le L de ETL) dans un ED. Le Framework permet de décrire l'ensemble des activités : celles effectuées avec succès et celles ayant échoué. Les sources considérées sont de type relationnel. Les données sont chargées dans un ED relationnel. Un algorithme est fourni permettant d'identifier et de filtrer les tuples pertinents des sources de données qui doivent être rechargés lors d'un échec de chargement.

Raman et al. [157] proposent une approche de transformation (le T de ETL) des données issues de différentes sources pour leur chargement dans un ED. Les données traitées par l'approche

sont des données tabulaires (fichiers ou bases de données). L'approche est supportée par un outil nommé Potter's Wheel. L'outil permet de spécifier les transformations graphiquement de façon itérative et interactive, dans le but de nettoyer les données. Les transformations sont basées sur des opérations algébriques appliquées sur un ensemble de données telles que les fonctions de changement de format, la suppression et l'ajout de colonnes, la division des colonnes selon un prédicat et la sélection selon une condition. L'outil est interactif dans le sens où l'utilisateur peut annuler les transformations non validées. La détection des anomalies et incohérences se fait automatiquement en arrière-plan sur la dernière vue des données transformées. Le chargement des données n'est pas traité dans l'article.

Vassiliadis et al. [206] proposent de définir le processus ETL par un modèle logique. Les auteurs modélisent le flux de données provenant des sources vers l'ED via une composition d'activités. La combinaison d'activités ETL et des sources de données définit un scénario ETL. Les scénarios sont ensuite représentés par un graphe appelé *graphe d'architecture*. Les activités ETL et les sources de données sont représentées par des nœuds dans le graphe. Ces nœuds sont liés par quatre différents types de relations (instance-de, partie-de, relation de régulateur et relation de fournisseur). Des règles de transformation permettant de réduire la complexité du graphe sont définies. Les schémas (sources et destination) considérés sont définis par une structure générique couvrant divers types de schémas logiques (BD relationnelles, fichiers plats, fichiers COBOL, cubes dimensionnels). Les auteurs déroulent cependant leur approche uniquement sur les bases de données relationnelles. Les flux de données également prennent en compte des aspects de la modélisation logique (comme le mapping des clés primaires et étrangères, les jointures, etc). Des algorithmes, basés sur les graphes des schémas, sont proposés. Ces algorithmes permettent de traiter l'extraction et le chargement des données. Certains aspects de transformations (filtrage, conversion, agrégation) ne sont pas traités.

Vassiliadis et al. proposent dans [205] un modèle conceptuel décrivant les activités du processus ETL. Le modèle définit les relations de mapping entre les sources et l'entrepôt ainsi que les transformations nécessaires au chargement des données dans l'entrepôt. La définition des caractéristiques du modèle conceptuel est décrite par un ensemble d'étapes constituant une méthodologie de modélisation conceptuelle du processus ETL. Cette méthodologie est complétée et enrichie dans [177]. Ce modèle conceptuel est défini dans le cadre d'une architecture composée d'une couche méta-modèle qui contient le modèle conceptuel générique et de la couche Template présentant une palette de constructeurs spécifiques aux processus ETL. Ces constructeurs sont représentés comme des sous-classes des classes de la couche méta-modèle (relation is-a). Ils sont obtenus par une relation de spécialisation (par exemple, l'agrégation est une sous-classe de la classe générique Transformation). Le modèle conceptuel générique est défini par un diagramme de classe. Il a pour but de fournir une documentation précise décrivant les caractéristiques des sources de données et de l'ED. Les auteurs estiment que cette documentation est nécessaire dès les premières phases du projet d'entreposage.

Trujillo et al. proposent dans [193] une approche de modélisation conceptuelle des processus ETL, via un diagramme de classe UML. Chaque activité ou mécanisme ETL (exemple. agregation, conversion, chargement, union) est représenté par un nouveau *stéréotype* dans le diagramme. Les stéréotypes sont accompagnés par une définition formelle sous forme de contraintes OCL. Des notes UML sont utilisées pour l'explication des fonctionnalités définies et la représen-

tation des mappings entre les attributs des sources et ceux de l'ED cible. Le but du modèle est de fournir une documentation facilitant la gestion de l'ED et sa maintenance. Un des principaux objectifs du modèle serait ainsi de pouvoir estimer l'impact de tout changement intervenant au niveau des sources de données sur l'ED.

Luján-Mora et al. proposent dans [127] de modéliser les mappings entre les attributs des sources et ceux de l'ED au niveau conceptuel en utilisant la notation UML. Les auteurs étendent l'approche citée précédemment [193], par un mécanisme de représentation des différents mappings entre les attributs des sources et les attributs du schéma cible de l'ED. Le modèle proposé représente les *attributs* des sources et de l'ED comme éléments de base de la modélisation. Le concept d'attribut peut être instancié par les concepts colonne, propriété ou champs. Le modèle de mapping proposé est basé sur la définition de stéréotypes UML. Les attributs peuvent être liés les uns aux autres par des associations avec le stéréotype *Map*. Les mappings sont définis en utilisant des packages avec stéréotype *Mappings*. Le concepteur peut zoomer sur les packages à différents niveaux de granularité afin de définir les détails jusqu'au niveau des attributs.

Luján-Mora et al. proposent dans [125] une approche de modélisation physique d'un ED et des processus ETL, en adaptant les digrammes de *composant* et de *déploiement* de la notation UML. Les deux diagrammes doivent être conçus conjointement par le concepteur et l'administrateur de l'ED. Le but est de permettre d'anticiper les décisions de conception physique liées à l'implémentation et au déploiement de l'ED dès le début du projet d'entreposage. La structure physique du processus ETL est modélisée par un diagramme nommé *Integration Transportation Diagram* (ITD). Ce dernier permet de lier les schémas physiques des sources de données avec le schéma physique de l'ED. Ce diagramme permet de décrire différents aspects d'implémentation physique du processus ETL comme les serveurs utilisés, les protocoles de communication, les SGBD utilisés, etc.

Mazon et al. proposent dans [135, 133] une approche de développement d'un ED et du processus ETL en appliquant le standard Model Driven Architecture (MDA) qui est une démarche de réalisation de logiciels proposée et soutenue par le groupe Object Management Group (OMG). L'approche MDA consiste à élaborer différents modèles, en partant d'un modèle métier abstrait (Computation Independent Model CIM), puis sa transformation en un modèle indépendant de la plateforme (Platform Independent Model PIM) et enfin la transformation de ce dernier en un modèle spécifique à la plateforme d'implémentation cible (Platform Specific Model PSM). Les phases de développement d'un ED sont alignées avec ceux de la démarche MDA comme suit : (1) un modèle CIM est développé pour identifier et décrire les besoins des utilisateurs. (2) Les CIM sont traduits manuellement vers des PIM représentés via des modèles de la notation UML. Chaque PIM correspond à un modèle conceptuel d'une phase de l'ED (un modèle PIM pour représenter les sources de données, un PIM pour le processus ETL, un modèle PIM pour l'application et un autre PIM pour la personnalisation de l'ED). (3) Chaque PIM est transformé automatiquement en un PSM selon la plateforme de déploiement. Ces transformations sont spécifiés formellement en utilisant un standard QVT (Query / View / Transformation). (4) Le code de la mise en œuvre de l'ED est obtenu à partir des PSM. La démarche globale est décrite de manière générale. La modélisation et le passage entre les différents modèles (CIM - PIM - PSM) du processus ETL sont simplement évoqués.

Simitsis et al. proposent dans [174, 178] une approche semi-automatique permettant le

passage d'un modèle conceptuel décrivant le processus ETL (conçu selon l'approche présentée dans [177]) vers un modèle logique ETL (conçu selon l'approche présenté dans [204]). L'approche est structurée autour des étapes suivantes : (1) raffiner le modèle conceptuel ETL, éliminer les ambigüités existantes et identifier les sources candidates actives, (2) faire correspondre les concepts et attributs du modèle conceptuel aux structures de données et attributs du modèle logique, (3) appliquer un algorithme de mise en correspondance des transformations du modèle conceptuel avec les activités du modèle logique ETL et identifier l'ordre d'exécution des activités, (4) enrichir le modèle logique avec les contraintes ETL imposées sur les données, (5) s'assurer de la cohérence de l'ordre des activités ETL, et enfin procéder à la génération du schéma représentant les séquences des activités ETL qui représente le schéma logique ETL.

Vassiliadis et al. proposent dans [204] une extension de leur précédente contribution [206, 203], et définissent un Framework de conception logique des activités ETL. Le Framework se base sur un métamodèle décrivant une activité ETL. Le métamodèle est conçu de manière générique pour capturer plusieurs types d'activités ETL. Il peut être instancié pour représenter un workflow d'activités ETL relatives à un domaine précis. Les activités ETL (transformations, filtrage, etc) sont formellement décrites en utilisant un langage LDL (Logical Data Language) qui est une variante du langage Datalog. Le code d'exécution de chaque activité est décrit par une déclaration LDL. Les activités définies d'une manière abstraite au niveau de la couche logique, sont matérialisées et exécutées par des modules logiciels spécifiques dans la couche physique. L'approche ETL est mise en œuvre par un outil *ARKTOS II* [173].

Skoutas et al. proposent dans [180] une approche de conception ETL basée sur une ontologie OWL et les annotations sémantiques, afin de spécifier formellement et explicitement la sémantique des schémas sources et cible de l'ED. Le but de l'approche est d'automatiser le processus ETL. L'approche s'effectue selon les étapes suivantes : (1) la construction d'un vocabulaire commun qui traite le domaine d'application. Le vocabulaire et les annotations sont construits par le concepteur à travers la lecture des modèles conceptuels des sources et de l'ED. (2) La proposition d'une méthode d'annotation des sources de données et de l'ED en utilisant le vocabulaire construit. Les sources et l'ED sont représentés par leurs schémas relationnels. Le modèle d'annotation proposé permet d'établir des mappings entre les schémas relationnels et les concepts du vocabulaire défini. (3) La proposition d'un algorithme de génération d'une ontologie d'application qui décrit le domaine d'application et les mappings entre cette ontologie et les schémas des sources et de l'ED. (4) La proposition d'un algorithme d'identification des transformations ETL. Les auteurs présentent une liste d'opérateurs habituellement rencontrés dans le processus ETL (exemple. Retrieve, Aggregate, Extract, etc). Une fois les transformations identifiées, les auteurs proposent d'utiliser l'approche présentée dans [174] pour définir la séquence des activités ETL.

Cette approche proposée dans [180] considère uniquement des sources de données relationnelles. Les auteurs étendent l'approche dans [181] en considérant les données structurées (relationnelles) et semi-structurées représentées en XML. Cette nouvelle approche suit les étapes suivantes : (1) la représentation des schémas des sources et de l'ED sous forme d'un graphe appelé *graph datastore* permettant d'unifier la représentation des données structurées et semi-structurées. (2) La représentation d'une ontologie d'application (*ontology graph*) créée par le concepteur et modélisée sous forme d'un graphe. Les classes de l'ontologie sont représentées par

les nœuds du graphe et les propriétés par ses arêtes. (3) Les graphes représentant les sources et l'ED sont annotés manuellement par le concepteur avec des labels (étiquettes) assignés aux nœuds des graphes. Ces annotations représentent les mappings entre les schémas des sources, l'entrepôt et l'ontologie d'application. (4) Sur la base du graphe d'ontologie et des graphes annotés représentant les sources et l'ED, un algorithme ETL est défini permettant d'identifier les sources de données pertinentes pour le peuplement de l'ED, et de définir les opérations ETL conceptuels pour l'intégration des données dans l'entrepôt cible. La traduction du processus ETL vers un modèle logique et son exécution ne sont pas présentés. Les auteurs étendent cette dernière approche dans [176, 175] en proposant d'exploiter l'ontologie afin de fournir une description textuelle (en langue naturelle) des résultats de la phase de conception ETL, à savoir les annotations des sources de données et le scénario ETL généré. Cette description textuelle permet de faciliter la validation de la phase ETL par les utilisateurs et les concepteurs. Cette approche s'effectue en trois étapes : (1) le prétraitement des sources qui consiste à extraire et à sélectionner des termes des sources de données afin de produire une terminologie commune. (2) La deuxième étape consiste à utiliser cette terminologie pour construire une ontologie décrivant les spécifications conceptuelles du processus ETL. (3) La dernière étape consiste à générer des rapports sur le processus ETL et les sources de données

Tziovara et al. proposent dans [196] une approche pour déterminer la meilleure configuration d'implémentation physique d'un workflow ETL. L'approche prend comme entrée la représentation logique du processus ETL et un modèle de coût. Le terme *enregistrement* est utilisé pour désigner un type d'enregistrement des données dans un schéma logique (relationnel ou fichiers). Une *activité* désigne tout type de module logiciel qui traite des données fournies en entrée, soit en effectuant une transformation des données ou en appliquant des procédures de nettoyage des données. Les activités et les jeux d'enregistrements sont des abstractions logiques des entités physiques. Un algorithme permettant la traduction de la représentation logique du workflow ETL (basé sur un graphe orienté acyclique) vers son modèle physique est défini. Cet algorithme est basé sur une bibliothèque de modèles réutilisables (template) pour les activités ETL logiques et physiques. Les modèles (ou templates) logiques matérialisent les activités logiques, et les modèles (ou templates) physiques matérialisent les activités physiques. Les modèles logiques et physiques sont ensuite mis en correspondance. Plusieurs alternatives de génération d'une implémentation physique du workflow ETL sont étudiées. Un modèle de coût générique est utilisé comme critère discriminatoire entre les représentations physiques.

Nebot et al. proposent dans [143, 144] une approche ontologique semi-automatique permettant la génération d'un ED peuplé avec des données du Web qui sont annotées par une ontologie. Les données considérées sont des instances OWL/RDF représentées et stockées sous forme de triplets (sujet, prédicat, objet). L'architecture de la solution proposée est définie en trois étapes : (1) la conception du schéma multidimensionnel à partir d'une ontologie globale ; ce schéma est défini par le concepteur selon les besoins des utilisateurs qui sont formulés par des expressions en logique de description. (2) L'extraction des instances des faits selon le schéma multidimensionnel établi précédemment. (3) L'extraction des instances des dimensions qui est réalisée sur la base de la table de faits générée et les relations de subsomption existantes dans l'ontologie. Les relations transitives permettent d'identifier les hiérarchies de dimensions. (4) La génération du cube de données multidimensionnel en appliquant les opérations d'agrégations récupérées via les requêtes

des utilisateurs.

Bergamaschi et al. proposent dans [21] une approche ETL exploitant un thésaurus pour faciliter l'alimentation de l'ED. Les schémas des sources et de l'ED sont préalablement définis au niveau logique relationnel. Un thésaurus décrivant les schémas des sources est défini et exploité pour générer les mappings entre les schémas des sources et le schéma de l'ED. Ces mappings sont basés sur des mesures de similarité sémantiques. Une fonction de transformation est définie pour gérer les conflits entre les sources et alimenter le schéma de l'ED.

Romero et al. proposent dans [166] une approche ontologique semi-automatique appelée *GEM*, pour la génération d'un modèle conceptuel multidimensionnel et la représentation conceptuelle des processus ETL. L'approche prend en entrée : (1) les données sources structurées (données relationnelles) et semi-structurées (données au format XML) représentées sous la forme d'une ontologie OWL. Cette ontologie est annotée par les entités des sources. L'ontologie est obtenue selon l'approche citée précédemment proposée dans [181]. (2) Les besoins des utilisateurs sont exprimés sous la forme d'un fichier XML structuré. Les balises du fichier XML représentent les concepts multidimensionnels (faits, dimensions et attributs). Les étapes de la solution proposée sont décrites comme suit : (1) la validation des besoins qui consiste à identifier les besoins qui sont en concordance avec les sources de données disponibles. Le fichier XML des besoins est analysé. Les concepts de l'ontologie correspondant à chaque besoin sont annotés par des concepts multidimensionnels, par le concepteur. (2) Identification des opérations ETL : cette étape comprend trois tâches : identifier les opérations nécessaires aux mappings des sources de données avec l'entrepôt cible ; vérifier et enrichir ces mappings par le concepteur en considérant de nouvelles fonctions issues des besoins non fonctionnels (recouvrement, rafraichissement, déduplication, etc) ; et rajouter des opérations de raffinement par exemple en traitant les dimensions dont les données changent rarement (slowly changing dimensions SCDs). (3) Réconciliation : L'approche génère un certain nombre de modèles multidimensionnels qui doivent être réconciliés en un unique modèle conceptuel multidimensionnel. Un processus conceptuel ETL est défini pour peupler chaque modèle multidimensionnel.

2.3.2 Analyse et comparaison des travaux de conception ETL de l'ED

Une des principales tâches d'un ED consiste à intégrer des données provenant de sources hétérogènes pour alimenter l'ED. Cette intégration se fait durant la phase de conception ETL. Par conséquent, plusieurs recherches ont été proposées exclusivement dédiées à cette phase de conception. Un ED est, de ce point de vue, considéré comme un *système d'intégration* matérialisant les données des sources selon un *processus ETL*. Un système d'intégration est défini par le triplet <G,S,M> [120] décrivant un schéma global (G), les sources de données (S) et les mapping entre les différents schémas (M). Ainsi, nous proposons de classifier les travaux traitant la phase ETL selon les quatre critères suivants (figure 2.4) : le *schéma global*, les *sources* considérées, les *mapping* entre le schéma global et les schémas des sources et le *processus ETL*. Nous étayons cette classification par des critères d'ordre général tels que : l'*automaticité* de l'approche proposée, le *déploiement* de l'ED et la *couverture du cycle* de conception par l'approche.

FIGURE 2.4 – Critère de classification des travaux de conception ETL de l'ED

2.3.2.1 Le schéma global

Nous analysons le schéma global sur lequel repose le processus d'intégration selon deux critères : le *niveau d'abstraction du schéma* et son *interprétation*.

Le premier critère indique le *niveau d'abstraction du schéma global* qui peut être : conceptuel [36, 135, 181, 144, 166], logique généralement sous forme d'un schéma relationnel [196, 204, 205, 127] ou physique [108, 157].

Le deuxième critère indique l'*interprétation intensionnelle* (définie au niveau *modèle*) ou *extensionnelle* (définie au niveau *instances*) du schéma global. Certains travaux considèrent uniquement le modèle du schéma global. Par exemple, *Calvanese et al.* [36] considèrent les concepts d'un schéma E/A enrichi. D'autres travaux considèrent les instances du schéma global comme [114].

2.3.2.2 Les sources considérées

Nous analysons les sources utilisées dans les différents travaux selon trois critères : le *type* de sources considérées, le *niveau d'abstraction* des schémas des sources et leur *interprétation*.

Le premier critère indique si les sources utilisées sont *classiques monocouches* (généralement relationnelles) [125, 196, 204, 36, 206], ou *conceptuelles multicouches*. Les sources conceptuelles peuvent être semi structurées [181, 166] ou ontologiques [144, 23].

Le deuxième critère indique le *niveau d'abstraction* des schémas des sources qui peut être : conceptuel [135, 144], logique généralement sous forme de schémas relationnels [206, 127, 174, 196] ou physique [108].

Le dernier critère indique si les sources sont considérées au niveau *extensionnel* [186, 35] ou *intensionnel* comme c'est le cas pour la plupart des travaux.

2.3.2.3 Les mapping

Les mappings définis pour faire correspondre les schémas des sources au schéma global sont analysés selon quatre critères : la définition d'un *modèle de mappings*, le *niveau d'abstraction* des mapping, la *relation sémantique* dénotée par les mappings et *le sens* du mapping.

Le premier critère indique si un modèle de mappings a été défini dans l'étude proposée (O pour oui, N pour non). Les modèles de mapping proposés décrivent les correspondances entre les entités des schémas des sources et les entités du schéma global.

Le *niveau d'abstraction* des mapping peut être : (1) complètement conceptuel où les mapping sont définis entre les schémas conceptuels des sources et le schéma conceptuel global, (2) des mapping partiellement conceptuels où l'un des schémas (global ou sources) est défini au niveau conceptuel, (3) des mappings logiques définis entre des schémas logiques (global et sources), (4) des mappings physiques définis entre des schémas physiques (global et sources).

La *relation sémantique* indique trois principaux types de mapping qui peuvent être utilisés :

- mappings d'équivalence : indique que les éléments liés par les mappings présentent la même sémantique dans le domaine décrit.
- mappings d'appartenance : indique que le premier élément du mapping est plus spécifique que le deuxième élément (et vice versa).

– mappings de chevauchement : indique que les deux éléments peuvent partager une sémantique commune.

Ce critère est généralement ignoré dans les différents travaux, ce qui nous laisse supposer que le mapping utilisé est le mapping par défaut d'équivalence. D'autres travaux traitent tous ces types de mappings [180, 181, 23, 18].

Le *sens du mapping* peut être de trois types : Local as View (LaV), Global as View (GaV) ou GLaV. Le sens du mapping indique si les concepts des schémas des sources sont définis comme des vues sur le schéma global (LaV) [36], ou si les concepts du schéma global sont définis comme des vues sur les schémas des sources (GaV) [206], ou si les mappings sont définis entre des vues du schéma global et des vues des sources (GLaV) [79].

2.3.2.4 Le processus ETL

Le processus ETL est analysé selon les critères suivants : l'*algorithme* ETL proposé, la *formalisation et implémentation* de l'algorithme ETL, la proposition d'un *modèle* ETL.

Le premier critère indique quelles sont les *activités ETL étudiées* (extraction, transformation ou chargement des données). Certains travaux proposent de traiter une partie du problème ETL, comme la transformation (T) [157] ou le chargement des données (L) [114]. La majorité des travaux proposent des solutions couvrant les trois activités du processus ETL.

Le deuxième critère indique si une *formalisation* du processus ETL et son implémentation sont fournies. La formalisation consiste à fournir l'algorithme ETL et l'implémentation consiste à tester l'algorithme sur un benchmark ou sur un cas réel.

Le troisième critère indique si le processus ETL est accompagné d'un *modèle ETL*. L'*objectif* de modéliser le processus ETL peut être de décrire l'ensemble des données, concepts et propriétés intervenant dans le processus ETL et les liens entre eux. D'autres modèles ont pour but de décrire le flux de données des sources jusqu'au schéma de l'ED cible [206, 204, 196, 53].

Les modèles fournis pour la modélisation des scénarii ETL peuvent être définis à différents *niveaux d'abstraction* : (1) au niveau physique en utilisant des diagrammes modélisant des aspects d'implémentation comme les diagrammes de composant et de déploiement [125]. (2) Au niveau logique comme les travaux de [186, 206, 204, 174, 178] qui modélisent le processus ETL par des graphes liant les tables ou enregistrements des sources aux tables de l'ED. (3) Au niveau conceptuel comme les travaux [186, 205, 177, 193] qui proposent de modéliser le processus ETL par un diagramme de classe. Certains travaux accompagnent leur modèle par une interface graphique pour la création manuelle des scénarii ETL [206].

2.3.2.5 Aspects généraux

Les aspects généraux portent sur les critères suivants :

2.3.2.5.1 L'automaticité de l'approche

Ce critère indique si la phase ETL est menée selon une approche manuelle, semi-automatique ou automatique. Les approches *automatiques* sont parfois accompagnés d'outils ETL comme [204, 157]. La plupart des approches sont *semi-automatiques* et nécessitent l'intervention du concepteur

généralement pour les tâches de définition des mappings et l'assignation de la sémantique aux sources.

Nous notons que très peu de travaux ETL dans la littérature proposent des expérimentations validant le processus ETL, mais se contentent de dérouler leur proposition sur un exemple proposé. Les quelques travaux proposant une évaluation de leur approche sont les suivants : *Romero et al.* [166] qui proposent des expérimentations évaluant la couverture des requêtes par l'ED obtenu après intégration par un processus ETL défini. *Nebot et al.* [144] proposent d'évaluer leur approche ETL en étudiant la performance en temps de chargement et la scalabilité de l'algorithme ETL.

2.3.2.5.2 Le déploiement de l'ED

Ce critère indique si le déploiement de l'ED se fait selon une architecture *monocouche* généralement relationnelle, comme c'est le cas pour la plupart des travaux ou selon une architecture *multidimensionnelle* sous forme d'un cube [144, 135] ou selon une architecture *multicouche* de base de données ontologique [104, 143].

2.3.2.5.3 La couverture du cycle de conception par l'approche

Ce dernier critère indique si tout le cycle de conception est *couvert* par l'étude. La plupart des travaux traitent les phases de conception du schéma de l'ED (conceptuel, logique, physique), séparément de la phase ETL. Quelques récents travaux peu nombreux proposent une approche conjointe couvrant les phases de conception et ETL [166, 144].

Les tableaux 2.3 et 2.4 synthétisent les principaux travaux cités et les analysent selon ces critères de comparaison. Dans ces tableaux, la mention "NM" signifie "critère défini mais non mentionné", et la mention "-" signifie "critère non défini". Nous étudions, juste après, le positionnement de notre deuxième contribution par rapport à l'ensemble de ces travaux selon les critères retenus.

2.3.3 Positionnement de nos contributions

La deuxième contribution de notre approche consiste à élever la conception ETL à la phase conceptuelle (plus particulièrement au niveau ontologique) afin de permettre un déploiement multiple de l'ED, selon les besoins des utilisateurs et décideurs. Cette phase de modélisation a été introduite dans le cycle de conception des ED de manière arbitraire pour répondre à la problématique d'intégration et de chargement des données dans un cycle de conception qui ne comprenait que les deux principales phases de modélisation logique et physique. Par conséquent, nous remarquons trois principales catégories d'approches ETL qui se dégagent, selon le niveau d'abstraction : les approches orientées niveau physique, les approches orientées niveau logique et les approches orientées niveau conceptuel. La dernière colonne du tableau 2.4 récapitule le positionnement de notre deuxième proposition par rapport aux travaux cités.

2.3.3.1 Approches ETL orientées niveau physique

Ce type d'approches procède à la définition du processus ETL au niveau physique, par la proposition de méthodes orientées algorithmes implémentés [114, 157, 196, 192] ou par la définition

Travaux/Critères	[108]	[36]	[186]	[114]	[157]	[206]	[205]	[193]	[127]	[125]
Schéma global										
Niveau abstraction	P	C	C	L	P	L	L	L	L	P
Interprétation	Int	Int	Int	Ext	Int	Int	Int	Int	Int	Int
Sources										
Sources considérées	Cl	Cl	Cl	Cl	Cl	Cl	Cl	Cl	Cl	Cl
Niveaux d'abstraction	P	L	P	L	P	L	L	L	L	P
Interprétation	Ext	Int	Ext	Ext	Ext	Int	Int	Int	Int	Int
Mappings										
Modèle de mappings	N	N	O	N	N	O	O	N	O	N
Niveau abstraction	P	PC	PC	NM	NM	L	L	L	L	L,P
Relation sémantique	NM	Eq	NM	NM	NM	NM	NM	NM	NM	NM
Sens mapping	NM	LaV	GLaV	NM	NM	GaV	NM	NM	NM	NM
Processus ETL										
Algorithme (E, T ou L)	ETL	ETL	ETL	L	T	ETL	ETL	ETL	ETL	-
Formalisation et génération du code	N	O	N	O	O	O	N	N	N	N
Modèle ETL										
Proposition modèle	N	N	O	N	N	O	O	O	O	O
Objectif de modélisation	-	-	D,F	-	-	F	D	D	D	D
Niveau abstraction	-	-	C	-	-	L	C	C	C	P
Généralités										
Automaticité	M	SA	M	A	A	SA	M	M	M	M
Déploiement	R	R	M	R	R	R	R	-	-	R
Couverture du cycle	O	N	N	N	N	N	N	N	N	N

TABLE 2.3 – Synthèse de comparaison entre les travaux de conception ETL de l'ED

de modèles physiques [125]. Ce type d'approches dépend fortement de la technologie de mise en œuvre choisie pour l'implémentation des flux d'intégration de données (comme les scripts personnalisés et le moteur ETL). Ces approches ignorent généralement les phases logique et conceptuelle car le modèle physique d'un processus ETL a un rôle important dans l'amélioration et l'optimisation des performances des applications décisionnelles. Les outils ETL proposés par l'industrie (comme *Microsoft Data Transformation Services*, *Oracle Warehouse Builder* ou *IBM. Data Warehouse Center*) et par la communauté de recherche cités précédemment [61, 140, 157] définissent la phase ETL au niveau physique. Ils considèrent le schéma global, les schémas sources considérés et les mappings au niveau physique.

2.3.3.2 Approches ETL orientées niveau logique

Ce type d'approches procède à la définition des différents processus ETL au niveau logique. Différents éléments peuvent être définis au niveau logique : le schéma global [114, 205, 193, 127], les schémas des sources [36, 114, 206, 193, 127], les mapping [205, 193, 127], ou le modèle fourni pour la modélisation des scénarii ETL [206, 204].

2.3.3.3 Approches ETL orientées niveau conceptuel

Nous remarquons que plusieurs des travaux proposés tentent d'élever certaines tâches de la phase ETL au niveau conceptuel. L'objectif de définir la phase ETL à un niveau d'abstraction plus

Travaux/Critères	[135]	[174]	[204]	[180]	[181]	[196]	[144]	[21]	[166]	[22, 18]
Schéma global										
Niveau abstraction	C	L	L	C	C	L,P	C	C	C	C
Interprétation	Int	Int	Int	Int	Int	Int	Int	Int	Int	Int
Sources										
Sources considérées	Cl	Cl	Cl	Cl	Cl/XML	Cl	Ont	Cl	Cl/XML	Cl/Ont
Niveaux d'abstraction	C	L	L	L	C	L,P	C	L	C	C
Interprétation	Int	Int	Int	Int	Int	Int	Int	Int	Int	Int
Mappings										
Modèle de mappings	O	N	N	N	N	N	N	N	N	O
Niveau abstraction	NM	L	L	PC	C	L,P	C	PC	NM	C
Relation sémantique	NM	NM	NM	Tp	Tp	NM	Eq	NM	Tp	Tp
Sens mapping	NM	NM	NM	LaV	LaV	NM	GaV	NM	LaV	GaV
Processus ETL										
Algorithme (E, T ou L)	-	ETL	ETL	ETL	ETL	ETL	ETL	ETL	ETL	ETL
Formalisation et génération du code	N	O	O	O	N	O	O	O	O	O
Modèle ETL										
Proposition modèle	O	O	O	N	N	O	N	N	N	O
Objectif de modélisation	D	D,F	F	-	-	F	-	-	-	D,F
Niveau abstraction	C	C,L	C	-	-	L,P	-	-	-	C
Généralités										
Automaticité	A	A	A	SA	SA	A	SA	A	SA	SA
Déploiement	R,M	R	R	R	R	R	M	R	R	Mu
Couverture du cycle	O	N	N	N	N	N	O	N	O	O

TABLE 2.4 – Synthèse de comparaison entre les travaux de conception ETL de l'ED

élevé peut être de : fournir une documentation des correspondances entre les sources et l'ED dès le début du projet d'entreposage [177], de faciliter l'interrogation des cubes multidimensionnels [186], de faciliter la maintenance de l'ED [193], ou d'automatiser le processus ETL [180].

Différents éléments peuvent être définis au niveau conceptuel : le schéma global, les schémas des sources, les mappings, ou le processus ETL. Certains travaux proposent ainsi de définir le schéma global au niveau conceptuel [36, 186, 135, 180, 181, 144, 21, 166] afin de fournir une description conceptuelle du domaine d'intérêt et une documentation précise. D'autres travaux plus récents proposent de définir les schémas des sources au niveau conceptuel (généralement par leur schéma E/A) [135, 181, 144, 166, 37]. Lorsque les deux types de schémas (global et source) sont définis au niveau conceptuel, il est cohérent de définir des mapping au niveau conceptuel afin d'assurer une indépendance totale de la phase d'implémentation [135].

Certaines méthodes proposent d'exploiter la sémantique d'une ontologie pour décrire le schéma global et/ou les schémas des sources, dans le but de faciliter le processus ETL. Ces méthodes *ontologiques* [181, 21, 166, 144] qui définissent la phase ETL au niveau conceptuel procèdent de diverses manières.

Skoutas et al. [181] proposent de définir le schéma global par une ontologie, les schémas des sources par une représentation conceptuelle sous forme de graphe. Des mappings sont définis également au niveau conceptuel. La traduction du processus ETL vers un modèle logique et son

exécution ne sont pas présentés.

Bergamaschi et al. [21] représente uniquement les schémas des sources par un thésaurus (qui peut être considéré comme une ontologie rudimentaire).

L'approche proposée par [166] traite la phase ETL uniquement au niveau conceptuel ontologique, mais aucune traduction vers le niveau logique et physique n'est présentée. L'approche génère un grand nombre de modèles multidimensionnels qui doivent être réconciliés. Ceci peut conduire à la génération de schémas inutiles et non exploités.

Dans [144], les auteurs optent pour une représentation conceptuelle du processus ETL, mais concentrent leur étude sur les données du web sémantique, fournies selon une représentation en *triplets* (sujet, prédicat, objet). Cette étude présente un effort d'abstraction au niveau conceptuel, du schéma global et des schémas des sources. Cependant, l'approche est orientée selon une seule représentation physique (triplet) qui reflète le stockage des données dans une base sémantique. De plus, les transformations de données utilisées dans cette approche correspondent uniquement aux opérations d'agrégation, les autres opérations de conversion et de filtrage ne sont pas abordées.

Ces deux dernières méthodes [166, 144], ainsi que deux autres méthodes [108, 133] allient quelques phases de conception et la phase ETL. Romero et al. [166] traitent la phase de modélisation conceptuelle et la phase ETL. Dans [108], l'ensemble des phases de conception et d'ETL sont traitées de manière informelle. Cette approche est considérée comme un guide détaillé de la conception des ED. Dans [133], les auteurs proposent une approche de développement d'un ED et du processus ETL en appliquant le standard Model Driven Architecture (MDA) ou différents modèles (à différents niveaux d'abstraction) sont définis pour soutenir les phases de conception d'un ED. Cependant, la phase ETL n'est pas détaillée, les différents modèles du processus ETL sont mentionnés mais ne sont pas fournis. Un inconvénient souvent cité pour ce type de méthode basé sur MDA est la difficulté d'application de ces méthodes dans la pratique en raison du manque d'aptitude des utilisateurs pour la compréhension des modèles et des règles formelles de transformation entre les modèles [72].

De plus, dans l'ensemble de ces méthodes considérant les phases de conception et d'ETL, les besoins des utilisateurs sont peu présents. Dans le cas où les besoins sont considérés, ils sont formalisés par des langages difficiles à appréhender pour l'utilisateur final (comme les logiques de description pour [144] et XML pour [166]). Ces deux formalisations ne sont pas familières aux utilisateurs ce qui ne facilitent pas la communication et la validation des besoins.

Nous remarquons ainsi qu'il y a eu différents efforts pour élever certaines tâches de la phase ETL au niveau conceptuel, mais aucune approche ne définit toute la phase ETL (G, S, M et processus ETL) au niveau conceptuel, ou ne déroule le processus ETL sur l'ensemble des phases de conception (logique et physique).

Notre deuxième proposition consiste à définir l'ensemble de la phase ETL au niveau conceptuel ontologique. Le schéma global, les schémas des sources, et les mappings sont définis au niveau ontologique. Nous fournissons un Framework de formalisation générique pour la représentation des sources de données qui sont candidates au processus ETL. Le processus ETL est défini par un algorithme basé sur des opérateurs conceptuels, qui sera traduit au niveau logique et physique sous différentes représentations, selon la demande des utilisateurs. Nous avons appliqué cette approche pour l'intégration des BDBO, qui sont des sources idéales pour une intégration ontologique.

2.4 Conclusion

Nous avons présenté dans ce deuxième chapitre les principaux travaux relatifs à la conception des ED. Nous avons distingué deux catégories de travaux : les travaux portant sur la conception du schéma de l'ED, et les travaux portant sur la conception ETL pour l'alimentation de l'ED. Pour chaque partie étudiée, nous avons commencé par présenter une synthèse des principaux travaux les plus référencés. Nous avons ensuite étudié ces travaux selon un ensemble de critères pertinents, qui nous ont permis de mieux analyser les travaux et de positionner nos contributions. Des tableaux comparatifs recensant l'ensemble des travaux selon les critères retenus sont présentés.

Plusieurs constats sont remarqués suite à l'analyse de ces travaux. Le premier constat confirme notre conclusion du premier chapitre, et démontre que la majorité des travaux de conception d'ED convergent vers le cycle de conception incluant les cinq phases de conception. Ces travaux étudient une ou plusieurs des phases du cycle. Quelques travaux récents proposent des méthodes conjointes couvrant des phases de conception et d'alimentation d'un ED.

Le deuxième constat démontre l'importance de la représentation des besoins des utilisateurs d'une part et de la sémantique du domaine d'autre part, lors de la conception des ED. Nous remarquons ainsi que la plupart des travaux de conception actuels convergent vers des approches de conception mixtes alliant les sources des données et les besoins des utilisateurs. Plusieurs aspects de la définition des besoins sont étudiés (leur représentation, leur spécification et l'objet d'analyse des besoins). Différentes méthodes de conception récentes convergent vers des approches de conception ontologique où la sémantique d'une ou plusieurs ontologies est exploitée pour expliciter les sources de données, pour spécifier les besoins des utilisateurs, ou pour identifier la structure multidimensionnelle du schéma de l'ED.

Le troisième constat révèle l'instabilité de la phase ETL dans le cycle de conception des ED. Nous remarquons que cette phase de conception a d'abord été considérée au niveau physique et logique. La phase ETL a plus récemment été considérée au niveau conceptuel afin de faciliter la documentation, la maintenance ou l'interrogation de l'ED.

Nous présenterons dans le reste de ce manuscrit nos deux principales propositions. Nous positionnons nos travaux comme une suite de consolidation du cycle de conception des ED. Notre première proposition renforce la représentation des besoins au sein de la structure d'un ED. Ce dernier est défini par une approche de conception ontologique. Notre deuxième proposition étudie la mise au niveau ontologique de la phase ETL au sein du cycle de conception des ED. Cette mise à niveau a pour principal objectif de libérer la conception des contraintes d'implémentation afin d'offrir un déploiement "à la carte" du schéma de l'ED.

Deuxième partie

Nos propositions

Chapitre 1

Vers un système d'ED manipulant données et traitements

Sommaire

1.1 Introduction

Nous avons montré, dans le premier chapitre, l'évolution du cycle de conception des ED vers un cycle incluant plusieurs modèles de données (au niveau conceptuel, logique et physique) et diverses plateformes de déploiement. Cette évolution implique plusieurs choix de conception pour le système de l'ED ; elle nécessite donc une communication étroite entre les acteurs du système dès le début du projet d'entreposage afin d'étudier les différentes alternatives de conception. Dans le deuxième chapitre, nous avons exposé les différents travaux de conception des ED proposés dans la littérature et nous avons constaté une tendance globale de ces travaux vers des approches de conception considérant les *besoins des utilisateurs* et incorporant davantage de *sémantique* lors du processus de conception. Cependant, nous remarquons que ces propositions ne répondent toujours pas aux exigences du cycle de conception évolué, qui nécessite une conception impliquant divers choix de modèles et de plateformes. Afin de s'adapter à ce cycle de conception, nous estimons qu'il est nécessaire de fournir une représentation prépondérante au sein de l'ED à la première ressource permettant la construction de l'ED : les *besoins des utilisateurs*.

Dans son ouvrage de référence [108], *Ralph Kimball* affirme que les besoins des utilisateurs influencent pratiquement toutes les décisions de conception et de mise en œuvre de l'ED. Cette vision met les besoins des utilisateurs au centre de la conception de l'ED comme illustré en figure 1.1. Selon *Kimball*, l'architecture de l'ED, son déploiement et sa maintenance doivent être axés sur les utilisateurs. Tous les choix de conception survenant lors des différentes étapes du cycle sont basés sur une compréhension des besoins et exigences des utilisateurs.

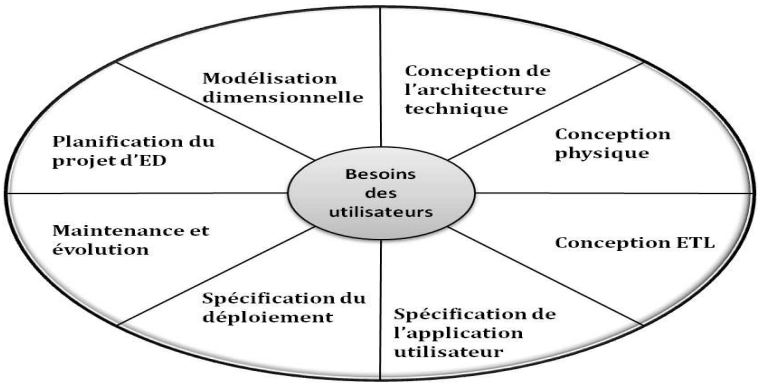

FIGURE 1.1 – Les besoins des utilisateurs influençant tous les aspects de l'ED [108]

Durant la première phase de conception de l'ED, les besoins des utilisateurs permettent d'identifier les données de l'ED représentées selon un modèle choisi et aussi l'ensemble des traitements auxquels devra répondre l'ED. Ces deux tâches (la structure de l'ED et ses traitements) doivent être menées en parallèle. Pour réaliser un produit d'ED autour de ce cycle, plusieurs corps de métiers interviennent : le rédacteur du cahier de charge, le concepteur, les représentants des éditeurs de SGBD, les programmeurs, les ingénieurs de logiciels, l'administrateur, le client et les utilisateurs finaux. Afin d'établir une communication étroite entre l'ensemble de ces acteurs, l'unique ressource disponible dès le début du projet d'entreposage consiste en les besoins des utilisateurs. La disponilibité des besoins permet d'effectuer diverses simulations des choix de conception offerts afin d'opter pour la meilleure conception possible. Notre principale proposition

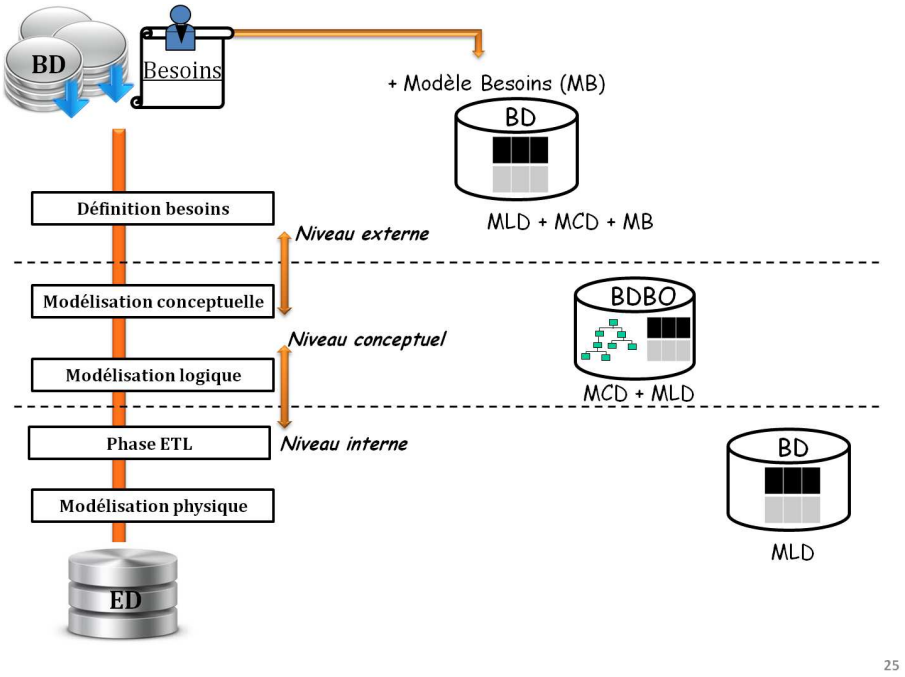

FIGURE 1.2 – Proposition 1 : persistance des besoins

dans ce chapitre consiste à faire persister les besoins des utilisateurs dans la structure finale de l'ED dans l'optique de fournir un processus de conception plus adapté aux évolutions du cycle de conception de l'ED (figure 1.2).

Nous concrétisons cette proposition en fournissant une méthode de conception du schéma de l'ED, dirigée par une ontologie de domaine et permettant de faire persister les besoins des utilisateurs dans la structure de l'ED. Le chapitre suivant portera sur l'alimentation de ce schéma par les données des sources selon un processus ETL. Pour notre première proposition, trois principales contributions sont présentées :

1. Proposition d'une méthode de conception du schéma de l'ED, spécifiant les sources de données et les besoins des utilisateurs *au niveau ontologique* [106, 99, 171]. Nous montrons l'intérêt de la définition des besoins des utilisateurs au niveau ontologique pour la spécification du modèle, sa modularité et le raisonnement sur le modèle des besoins.
2. Proposition d'une architecture d'ED faisant cohabiter les deux principaux modèles de l'ED : le modèle de *données* et le modèle de *besoins* [103, 104, 98, 105]. Nous proposons de représenter le schéma de l'ED dans une architecture BDBO qui fournit un mécanisme de persistance aux deux modèles de données logique et conceptuel. Nous étendons le méta-schéma ontologique de la BDBO par le modèle de besoins. Le schéma de l'ED sera validé sur des BDBO définies en utilisant deux SGBD sémantiques (le SGBD académique *OntoDB* et le SGBD sémantique industriel d'*Oracle*).
3. Expérimentations et tests : différentes expérimentations sont menées utilisant le banc d'essai SSB et qui permettent de simuler différentes tâches d'optimisation de l'ED à partir des besoins des utilisateurs [19, 102].

Ce chapitre est organisé en cinq sections comme suit. La section 2 présente les fondements et notions théoriques nécessaires à l'établissement et la compréhension de nos contributions. Nous exposons dans cette section les notions relatives aux ontologies et aux BDBO. Nous étudions l'influence des ontologies dans le domaine de l'ingénierie des besoins. Nous montrons (de manière théorique) l'impact et l'intérêt de la persistance des besoins sur le cycle de vie de l'ED. La section 3 présente la méthode de conception ontologique proposée. Nous commençons par énoncer les hypothèses nécessaires à la faisabilité de notre approche. Nous formalisons les entrées de notre méthode. Nous présentons ensuite les étapes de la méthode de conception proposée. La section 4 démontre la faisabilité et la validation de notre méthode. La dernière section conclut ce chapitre.

1.2 Fondements théoriques

Cette section décrit les principales notions nécessaires à la compréhension de notre proposition. Nous aborderons les principaux points suivants :

1. Nous commençons par présenter les notions relatives aux ontologies.
2. Nous analysons le rôle des ontologies dans la conception des BD et des BDBO.
3. Nous analysons ensuite le rôle et l'utilisation des ontologies dans le domaine de l'ingénierie des besoins (IB).
4. Nous étudions enfin l'impact de la persistance des besoins sur le cycle de vie de l'ED.

1.2.1 Notions sur les ontologies

Dans cette section, nous présentons les principales notions relatives aux ontologies. Nous définissons le concept d'ontologie, nous comparons les ontologies aux modèles conceptuels et nous présentons une classification des ontologies.

1.2.1.1 Définition d'une ontologie

Plusieurs définitions d'ontologies ont été proposées. Nous retenons la définition proposé dans [151] qui définit l'ontologie comme : *une représentation formelle, explicite, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine sous forme de classes, de propriétés et de relations qui les lient.* Les termes les plus importants dans cette définition sont donc :

- *Formelle* : l'ontologie est définie dans un langage traitable par machine.
- *Explicite* : l'ensemble des concepts et propriétés d'une ontologie sont spécifiés explicitement indépendamment d'un point de vue particulier ou d'un contexte implicite.
- *Référençable* : signifie que tout concept de l'ontologie peut être référencé de manière unique afin d'expliciter la sémantique de l'élément référencé.
- *Consensuelle* : signifie que l'ontologie est admise et acceptée par l'ensemble des acteurs d'une communauté.

1.2.1.2 Les notions clés dans une ontologie

Même si les modèles d'ontologies peuvent différer, ils reposent tous sur les mêmes notions qui sont : les *concepts* d'un domaine modélisés par les *classes* et leurs *attributs*, les *relations* entre

ces concepts, et les *axiomes*.

- *Le concept* : un concept peut se définir comme une entité composée de trois éléments distincts :
  - Le *terme* : exprimant le concept en langage naturel.
  - *Notion* ou *intension* du concept : la signification du concept.
  - Les objets dénotés par le concept, appelés également *réalisations* ou *extensions* du concept.

  Prenons l'exemple du concept *Voiture*. Le terme de ce concept est le nom commun *voiture*. Sa notion est d'être un véhicule de transport à roues. Son extension est l'ensemble des marques de voitures existantes (*Chevrolet*, *Kia*, etc).

  Les concepts d'une ontologie peuvent être classés en deux catégories [78] :
  - *Les concepts primitifs* : ce sont les concepts de base à partir desquels d'autres concepts de l'ontologie peuvent être définis. Ces sont des concepts dont l'ontologie ne fournit pas de définitions complètes. Leurs définitions reposent sur une documentation textuelle ou sur un savoir partagé par les membres d'une communauté.
  - *Les concepts définis* : ce sont les concepts définis par une définition complète, i.e par des conditions nécessaires et suffisantes exprimées en termes d'autres concepts (primitifs ou définis).
- *Les classes* : elles représentent le centre d'intérêt de l'ontologie et décrivent les concepts d'un domaine. Une classe peut avoir des sous-classes qui représentent des concepts plus spécifiques que la super classe (ou classe supérieure). Une classe peut avoir des instances représentant les extensions du concept. Par exemple, Chevrolet est une instance de la classe Voiture. Notons qu'une ontologie ainsi que l'ensemble des instances de toutes ses classes constituent une base de connaissances.
- *Les attributs* : Les attributs décrivent les propriétés des classes et des instances de l'ontologie.
- *Les relations* : désignent les associations définies entre les concepts de l'ontologie, comme la relation de subsomption *is-a* ou *est-un* (sous classe d'une classe).
- *Les axiomes* : désignent les assertions acceptées comme vraies dans le domaine étudié. Les axiomes et les règles permettent de vérifier la cohérence d'une ontologie, et aussi d'inférer de nouvelles connaissances. Par exemple, si deux personnes sont frères alors il existe une personne qui est la mère de chacun d'eux.

Les ontologies peuvent être classées selon leur objet de conceptualisation en plusieurs types [155] : ontologie de *représentation des connaissances*, ontologie *supérieure ou de haut niveau*, ontologie *générique*, Ontologie de *domaine*, ontologie de *tâche* et ontologie d'*application*. Nous nous intéressons aux *ontologies de domaine*, utilisées dans la conception des BD. Les ontologies de domaine sont définies comme des ontologies qui décrivent le vocabulaire ayant trait à un domaine particulier notamment en spécialisant les concepts d'une ontologie de haut niveau.

1.2.1.3 Domaines d'émergence des ontologies

Les ontologies ont été utilisées ces dernières années à différentes fins dans plusieurs domaines comme le traitement du langage naturel, l'interopérabilité des logiciels, le web sémantique, et la conception des bases de données [188].

Dans le domaine du *traitement automatique du langage naturel*, la sémantique apportée par les ontologies permet la résolution des conflits survenant lors de l'analyse syntaxique et sémantique d'un texte (polysémie, synonymie, etc).

Dans le domaine de *l'interopérabilité des systèmes*, les ontologies permettent de faire interagir des logiciels d'un même domaine. Dans un logiciel, un lien formel existe entre le code de l'application et le modèle à partir duquel est généré ce code. Grâce à l'aspect consensuel des ontologies, différents modèles d'un même domaine définis pour plusieurs logiciels peuvent être liés à une ontologie de domaine, ce qui permet de faire interagir ces logiciels. Cette approche est appelée "ingénierie logicielle dirigée par les ontologies" [188].

Dans le domaine du *web sémantique*, les ontologies peuvent être utilisées pour indexer les services disponibles sur le réseau Internet. Un même service peut être décrit par plusieurs noms différents attribués par des créateurs de services différents, ce qui rend difficile pour un utilisateur débutant ou même expérimenté la recherche de ces services. L'utilisation d'ontologies pour décrire ces services contribue à rendre la recherche automatisable et réalisable par des agents logiciels [188].

Dans le domaine des *bases de données*, les ontologies de domaine ont été utilisées pour :

- L'échange des données : une des caractéristiques des ontologies est qu'elle est référençable, ce qui signifie que tout concept de l'ontologie peut être identifié de manière *unique*. Ainsi, pour des BD référençant une ontologie, la signification de chaque élément d'information peut être définie localement en référençant des identifiants de concepts d'une ontologie, ce qui facilite considérablement l'échange de données entre ces bases.
- La conception des BD : une ontologie étant une conceptualisation d'un domaine de connaissances, elle peut être utilisée comme base de conception d'une BD. Dans le cycle de conception des BD et ED présenté au premier chapitre, nous avons montré que les ontologies ont étendu le cycle de manière *horizontale* au sein de la phase conceptuelle. Nous analysons dans le point suivant l'apport des ontologies dans une démarche de modélisation conceptuelle.

1.2.1.4 Ontologies et modèles conceptuels

Les ontologies présentent des similitudes avec les modèles conceptuels classiques sur *le principe de la modélisation* car tous deux définissent une conceptualisation de l'univers du discours au moyen d'un ensemble de classes auxquelles sont associées des propriétés [57]. Les ontologies et les modèles conceptuels divergent cependant sur un point essentiel : *l'objectif de modélisation* qui est à l'origine de la contribution d'une ontologie dans une démarche de modélisation conceptuelle. Une ontologie est ainsi construite selon une approche *descriptive*, par contre un modèle conceptuel est construit selon une approche *prescriptive*. Une approche prescriptive a plusieurs implications [187] :

- Seules les données pertinentes pour l'application cible sont décrites.
- Les données doivent respecter les définitions et contraintes définies dans le modèle conceptuel.
- Aucun fait n'est inconnu : c'est l'hypothèse du monde fermé.
- La conceptualisation est faite selon le point de vue des concepteurs et avec leurs conventions.

– Le modèle conceptuel est optimisé pour l'application cible.

Ces caractéristiques, dues au fait qu'un modèle conceptuel dépend fortement du contexte dans lequel il a été conçu, sont à l'origine des hétérogénéités entre les schémas des BD, qui rendent leur intégration difficile. Contrairement à un modèle conceptuel qui prescrit une base de données selon des besoins applicatifs (orienté *application*), une ontologie est développée selon une approche descriptive (orienté *domaine*). Elle permet de décrire les concepts et propriétés d'un domaine donné indépendamment tout objectif applicatif et de tout contexte hormis le domaine sur lequel porte l'ontologie.

D'autres caractéristiques des ontologies peuvent aussi être exploitées pour la spécification d'un modèle conceptuel :

– *Identification des concepts* : les concepts dans une ontologie possèdent des identificateurs universels (URI) leurs permettant d'être référencés par des applications externes. Cet aspect peut être utilisé pour faire communiquer des applications d'entreposage entre elles dans le contexte des entrepôts fédérés ou les ED peer to peer.
– *Consensualité* : l'aspect consensuel des ontologies facilite aux concepteurs, travaillant sur divers projets référençant les mêmes ontologies, le partage et l'échange de données sur leurs modèles.
– *Raisonnement* : les ontologies, de part leur aspect formel, offrent des mécanismes de raisonnement permettant de vérifier la consistance des informations et d'inférer de nouvelles données.

A la lumière de ces différences, il est par conséquent intéressant de considérer une ontologie comme premier niveau de spécification d'un modèle conceptuel. Nous citons comme exemple les trois travaux suivants proposés par Roldan-Garcia et al. [141], Sugumaran et al. [189] et Fankam et al. [57] ; qui permettent de définir le MCD d'une BD à partir d'une ontologie supposée préexistante.

Roldan-Garcia et al. [141] ébauchent une méthodologie de conception passant par deux étapes : (1) L'extension si besoin de l'ontologie de domaine par de nouveaux concepts et propriétés, nécessaires pour l'application à mettre en œuvre. (2) L'extraction d'un sous ensemble de classes de cette ontologie, représentant le modèle conceptuel de la BD.

Sugumaran et al. [189] propose une méthode et un outil d'aide à la conception reposant sur une ontologie linguistique. La méthode analyse les termes d'une expression issue du cahier des charges définie en langage naturel. En faisant référence à l'ontologie linguistique, la méthode suggère différents concepts à introduire dans le modèle conceptuel. Les auteurs proposent également d'ajouter à l'ontologie des contraintes d'intégrité permettant de valider le modèle conceptuel. Par exemple la contrainte ontologique *prerequisite* liant le terme acheteur au terme article à vendre. Si le MCD possède la notion d'acheteur et pas celle d'article, la méthode propose de compléter le modèle par une entité article.

Fankam et al. [57] propose une méthode de conception de BD nommée SISRO (Spécialisation, Importation Sélective et Représentation des Ontologies). Cette méthode se base sur la sélection d'une ontologie conceptuelle supposée existante couvrant le domaine de la BD à concevoir. Cette ontologie peut être étendue manuellement par de nouveaux concepts ou propriétés. Le concepteur sélectionne un sous ensemble de cette ontologie considéré comme le modèle conceptuel de la BD, qui sera traduit en modèle logique puis physique. Cette méthode s'inscrit dans la même démarche que la méthode proposée par Roldan-Garcia et al. [141], mais conserve en plus

les correspondances entre l'ontologie et le MCD généré.

De la même manière que pour les BD, les ontologies peuvent également être exploitées pour la conception des ED. Nous nous inspirons d'ailleurs des méthodes citées pour la méthode de conception du schéma d'ED que nous proposons et que nous présentons dans la section suivante (cf. section 1.3).

1.2.1.5 Taxonomie des ontologies

Avec l'émergence des ontologies dans plusieurs domaines, la notion d'ontologie a été utilisée par plusieurs communautés de recherche. Initialement utilisée par la communauté de l'*intelligence artificielle* pour la gestion des connaissances, les ontologies ont ensuite été exploitées dans le domaine de la *linguistique* et celui des *bases de données*.

L'usage des ontologies diffère d'une communauté à une autre selon la manière de conceptualiser un domaine. Dans le domaine de la *linguistique*, les ontologies décrivent et manipulent des mots et les relations entre ces mots tels que la synonymie et l'antonymie. Ces ontologies sont dites ***ontologies linguistiques (OL)***.

Dans les deux autres communautés, on s'intéresse aux ontologies dites *conceptuelles* qui manipulent des concepts et non des mots. Dans le domaine des *bases de données*, on s'intéresse aux concepts primitifs (qui ne sont pas définis par d'autres concepts). Ces concepts, possèdent une représentation unique. Ils sont utiles pour la conception des BD où les redondances sont exclues, et également dans les formats d'échange où il doit exister une seule représentation possible pour chaque information échangée. Les ontologies ne contenant que des concepts primitifs sont appelées ***ontologies conceptuelles canoniques (OCC)*** [151]. Par exemple, on peut considérer une OCC avec une classe Personne caractérisée par deux propriétés Nom et Sexe (masculin ou féminin).

Dans le domaine de *l'intelligence artificielle*, et afin d'avoir la possibilité de faire des déductions, on s'intéresse aux concepts définis. Ces concepts sont exprimés en utilisant d'autres concepts (primitifs ou définis) par des relations d'équivalence entre concepts. Ces relations peuvent être des relations entre classes appelées opérateurs de classes comme les opérateurs orientés ensembles (comme l'union, l'intersection et la différence) et les opérateurs de restrictions sur les valeurs de propriétés (comme les cardinalités des propriétés et les quantificateurs universel et existentiel appliqués sur des propriétés). Elles peuvent également être des relations entre propriétés, appelés opérateurs de propriétés, tels que les relations algébriques ou logiques (propriétés transitives, symétriques, etc) [146]. Les ontologies contenant des concepts primitifs et définis sont appelées ***ontologies conceptuelles non canoniques (OCNC)*** [151]. En reprenant l'exemple précédent du concept primitif Personne qu'on spécialise par les deux concepts définis Homme (Personne de sexe masculin) et Femme (Personne de sexe féminin), nous obtenons les concepts d'une OCNC.

L'usage fait des ontologies par ces trois communautés (sans vouloir dire que ces catégories d'ontologies sont réservées à une seule communauté particulière) permet ainsi de classer les ontologies selon la taxonomie proposée par [151] illustrée par le *modèle en couches* dans la figure 1.3. Ce modèle, appelé aussi *modèle en oignon* représente les trois catégories d'ontologies. La base de ce modèle est formée par les ontologies OCC qui fournissent une base formelle pour modéliser et échanger les connaissances d'un domaine. Une deuxième couche est formée par les OCNC,

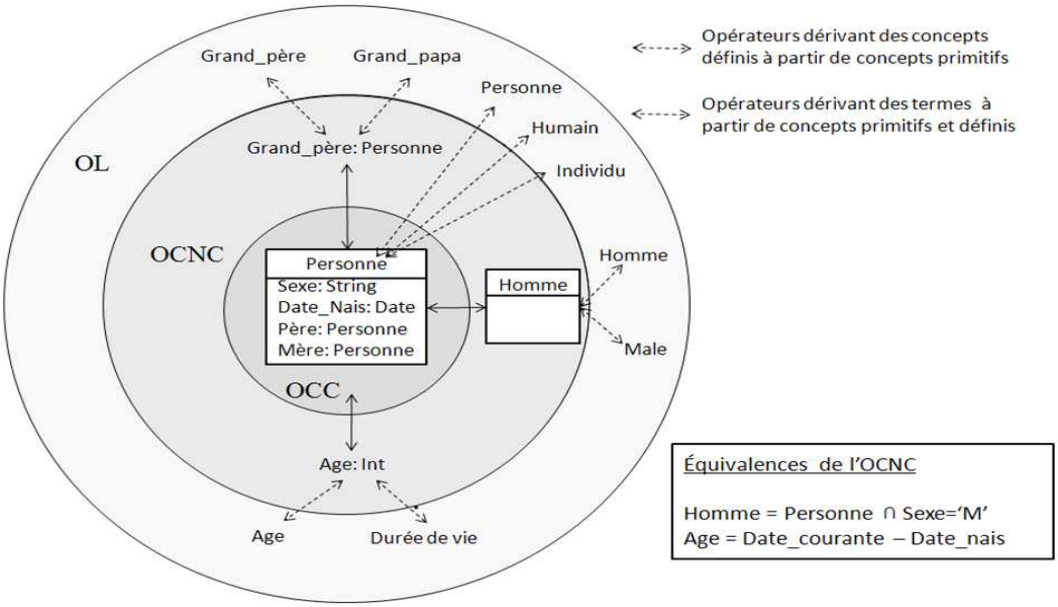

FIGURE 1.3 – Modèle en oignon [58]

qui fournissent des mécanismes permettant de représenter les connaissances d'un domaine par différentes conceptualisations. La dernière couche est formée par les OL qui fournissent une représentation des concepts d'un domaine en langage naturel éventuellement dans plusieurs langues possibles. Cette classification des ontologies en couches possède une dimension méthodologique, où deux méthodes complémentaires de conception d'ontologies peuvent être utilisées. Une première méthode "des mots aux concepts" consiste à extraire la couche conceptuelle de la couche linguistique. Cette méthode nécessite l'extraction semi-automatique des termes du domaine (à partir de documents textuels par exemple) et la définition des concepts à partir de ces termes.

La deuxième méthode "des concepts aux mots" consiste à extraire la couche linguistique à partir de la couche conceptuelle. Cette méthode nécessite la définition des concepts canoniques d'un domaine obtenus par un consensus d'experts, le choix des opérateurs d'équivalence entre les concepts pour définir les concepts non canoniques ou dérivés, et finalement la définition d'une vue langagière sur les concepts de l'ontologie.

L'utilisation extensive des ontologies dans plusieurs domaines et par plusieurs communautés a permis la généralisation de l'indexation à base ontologique. Un volume important de données référençant les ontologies a donc été créé. Ces nouvelles données sont appelées données sémantiques ou *données à base ontologiques* (DBO). Quelques systèmes ont été proposés pour gérer ce type de données en mémoire centrale comme *OWLIM* [111] ou *Jena* [137]. Ces systèmes facilitent le chargement et la mise à jour des données mais deviennent impraticables lorsqu'il faut gérer un grand volume de données ontologiques. Pour assurer le passage à l'échelle, de nombreux systèmes ont proposé de gérer les DBO au sein de bases de données qui proposent des mécanismes efficaces pour le stockage et l'interrogation des données. Cette nouvelle architecture de base de données est appelée *base de données à base ontologique* (BDBO).

1.2.2 Les bases de données à base ontologique (BDBO)

1.2.2.1 Définition d'une BDBO

Une base de données à base ontologique est définie comme une base de données permettant de spécifier la sémantique de ses données de manière explicite, ayant les caractéristiques suivantes [56] :

- Les données ainsi que l'ontologie définissant leur sémantique sont stockées au sein de la base de données, et peuvent être sujettes aux mêmes traitements (interrogation, insertion, suppression).
- Chaque donnée est associée à son référent ontologique et inversement chaque concept ontologique est associé aux données qui lui correspondent.
- L'ontologie locale à la base de données peut faire référence à une ontologie externe.

Le développement des BDBO a suscité l'intérêt des deux communautés industrielle et de recherche, et plusieurs systèmes de BDBO ont été proposés : *IBM SOR* [124], *Oracle* [215], *Rdfsuite* [9], *Ontodb* [56]. Le développement de nombreuses BDBO résulte principalement des points suivants :

1. *la diversité des formalismes ontologiques* : chaque BDBO utilise un formalisme particulier pour définir son ontologie. Plusieurs langages de manipulation des ontologies ont été proposés répondant à différents objectifs tels que : *PLIB*, *RDF*, *DAML*, *OIL*, *DAML+OIL*, *OWL*.
2. *la diversité des modèles de stockage* : contrairement aux bases de données traditionnelles, où le modèle logique est stocké selon une approche relationnelle, dans une BDBO, différents modèles de stockage sont utilisés pour stocker les deux niveaux de modélisation : le niveau ontologie et le niveau des instances ontologiques.
3. *la diversité des architectures cibles* utilisées par le système de gestion de bases de données : une BDBO peut utiliser un seul schéma commun ou différents schémas pour stocker l'ensemble des données.

Nous détaillons dans ce qui suit ces trois critères distinguant les différentes BDBO.

1.2.2.2 Les formalismes ontologiques

Plusieurs langages d'ontologies ont été développés ces dernières années. On distingue les modèles d'ontologies supportant la définition des concepts primitifs uniquement (OCC) comme le modèle PLIB [151] développé pour le domaine de l'ingénierie ; et les langages supportant la définition des concepts définis (OCNC) comme les langages du web sémantique *RDF*, *RDF Schema*, *DAML+OIL* et *OWL*.

- *PLIB* : ce langage a été conçu initialement afin de caractériser de manière précise les produits du domaine technique. Partant du constat qu'on ne peut définir de nouveaux termes qu'à partir de termes primitifs et que les termes primitifs d'un domaine technique sont eux-mêmes très nombreux et difficiles à appréhender ; l'objectif essentiel d'une ontologie PLIB est de définir de la façon la plus précise et la plus concise possible les catégories et les propriétés primitives qui caractérisent les objets d'un domaine du monde réel [151]. Ce modèle permet ainsi de créer des ontologies conceptuelles canoniques en utilisant les

constructeurs proposés. Une ontologie PLIB permet la définition de classes, de propriétés, de domaine de valeurs et d'instances, et permet également de définir la hiérarchisation des classes et des propriétés. L'ensemble de ces concepts sont identifiés de manière unique par un GUI (Globally Unique Identifier).

- *Resource Description Framework (RDF)* : l'information sur le Web étant compréhensible seulement au niveau lexical-syntaxique, les outils logiciels indépendants ne sont pas en mesure de traiter l'information en l'absence des méta-informations [4]. Ceci a amené le W3C (World Wide Web Consortium) à élaborer une couche supplémentaire au dessus de XML (eXtensible Markup Language) appelée Ressource Description Framework (RDF) pour traiter ces méta-informations. Le W3C a développé RDF, un langage d'encodage de la connaissance sur les pages Web pour rendre cette connaissance compréhensible par les agents électroniques qui effectuent des recherches d'informations. RDF permet de décrire des ressources simplement et sans ambiguïté. Toute ressource est décrite par des phrases minimales composées d'un sujet, d'un verbe et d'un complément. On parle alors de déclaration RDF.
- *RDF Schema ou RDF(S)* : le langage RDF n'introduit que très peu de prédicats prédéfinis, il ne propose pas de constructeurs pour la conception d'ontologies. Il a donc été étendu par de nouveaux constructeurs pour permettre la définition d'ontologies, ce qui a donné lieu au modèle RDF(S). RDF(S) est un des piliers du web sémantique car il permet de bâtir des concepts définis par rapport à d'autres concepts ayant la particularité d'être partagés à travers le web. RDF(S) définit la connaissance d'un domaine sous forme de classes, de sous classes, d'instances de classes et de relations entre les classes. Ce langage est cependant limité et ne permet que la description des vocabulaires simples (pas de cardinalités, deux classes ne peuvent pas avoir des instances communes, pas de liens précis entre les classes et entre les propriétés, etc). Pour des vocabulaires plus complexes, les langages DAML+OIL ou OWL ont été proposés.
- *DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer)* : DAML+OIL a été crée suite à la fusion de deux travaux d'équipes de recherche qui sont : DARPA Agent Markup Language (DAML) du ministère de la défense américain, et Ontology Inference Layer (OIL) développé par la communauté de recherche européenne. DAML est un langage de représentation qui fournit une sémantique formelle pour l'information. OIL a enrichi RDF(S) en offrant de nouvelles primitives permettant de définir les classes comme l'union de classes, l'intersection de classes et le complémentaire d'une classe. La fusion des deux langages a donné lieu à DAML+OIL, écrit en RDF, permettant d'enrichir le pouvoir d'expression de RDF(S). Le W3C a utilisé le DAML+OIL comme base pour la construction de son propre langage d'ontologies : le langage OWL.
- *OWL (Ontology Web Language)* : OWL est un langage de description d'ontologies conçu au départ pour la publication et le partage d'ontologies sur le web sémantique. Il définit un vocabulaire riche pour la description d'ontologies complexes. Issu des logiques de description, OWL est basé sur une sémantique formelle définie par une syntaxe rigoureuse. Il

4. Une méta-information est une information complémentaire d'une autre information permettant le traitement de cette dernière par un outil logiciel indépendant. L'auteur qui fournit la méta information est le même qui produit l'information traitée.

dispose de fonctions facilitant la réutilisation d'autres ontologies et permet de définir des rapports complexes entre les ressources. Il intègre divers constructeurs sur les propriétés et les classes comme l'identité, l'équivalence, le complément d'une classe, les cardinalités, la symétrie, la transitivité, la disjonction, etc.

Cette revue des différents langages d'ontologies nous a permis d'étudier les caractéristiques et spécificités de ces langages. Le langage d'ontologies sur lequel se base notre méthode de conception est le langage OWL. Ce choix est principalement motivé par le fait qu'OWL est un langage très répandu recommandé par le W3C, ayant un pouvoir expressif très riche permettant de définir des OCC et des OCNC et qui est rapidement devenu le standard le plus utilisé.

OWL se présente en trois sous langages à expressivité croissante : OWL Lite, OWL DL et OWL Full. OWL DL est défini comme une extension d'OWL Lite, et OWL Full comme une extension d'OWL DL.

- *OWL Lite* : est le langage le plus simple syntaxiquement. Il permet de définir des hiérarchies de classes sur lesquelles sont spécifiées de simples contraintes. Il est par exemple utilisé pour la migration de thésaurus existants ou la construction de simples hiérarchies.
- *OWL DL* : possède un pouvoir expressif plus important qu'OWL Lite. OWL DL garantit la complétude des raisonnements (toutes les inférences sont calculables) et leur décidabilité (leur calcul se fait en une durée finie) [GHO09].
- *OWL Full* : est le langage le plus complexe. Il offre une grande expressivité mais ne garantit pas toujours la complétude des raisonnements ni leur décidabilité.

Ces trois sous langages établissent un compromis entre leur pouvoir expressif et leur pouvoir de raisonnement. Plus le langage est expressif, moins il assure ses capacités de raisonnement. Notons également que toute ontologie OWL LITE valide est également valide en OWL DL, et toute ontologie OWL DL valide est une ontologie OWL Full valide. Nous considérons dans notre approche une ontologie de domaine décrite en langage OWL-DL, pour la complétude et la décidabilité de ses raisonnements.

1.2.2.3 Les modèles de stockage utilisés dans une BDBO

Contrairement aux bases de données traditionnelles où le modèle logique est stocké habituellement selon une représentation relationnelle, dans une BDBO, une variété de modèles de stockage sont utilisés pour stocker deux niveaux de modélisation : le niveau du *modèle de l'ontologie* et le niveau des *instances ontologiques*.

Une première approche utilisée pour la représentation des données dans une BD relationnelle est l'approche *table universelle* qui stocke toutes les propriétés dans une seule table universelle. Cette représentation n'est plus utilisée car elle montre de nombreux inconvénients (grand nombre de colonnes et plusieurs colonnes représentent des valeurs nulles). Trois approches répandues ont été suivies pour la représentation des ontologies dans une BDBO [2, 56] : l'approche *verticale*, l'approche *binaire* et l'approche *horizontale*.

L'approche **verticale** représente toutes les données (schéma et instances ontologiques) par une unique table de triplets composée des trois colonnes (Sujet, Prédicat, Objet). Ces trois colonnes représentent respectivement l'identifiant d'un élément de l'ontologie, un prédicat et la valeur du prédicat. Cette représentation facilite l'insertion de nouveaux triplets. La mise à jour d'une information peut être plus difficile si elle nécessite l'accès à plusieurs triplets, ce qui

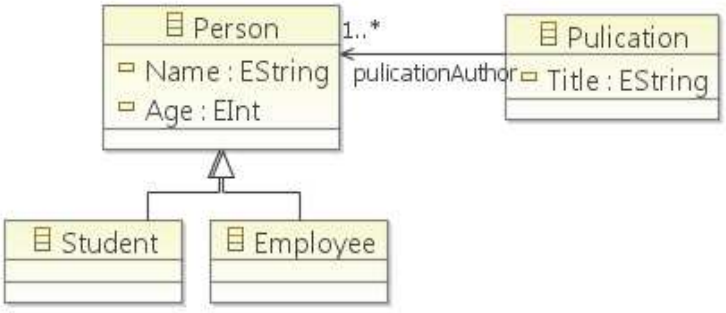

FIGURE 1.4 – Exemple de schéma d'une ontologie

TRIPLES		
Sujet	**Prédicat**	**Objet**
Stud#1	**Type**	**Student**
Stud#1	**Name**	**Name1**
Stud#1	**Age**	**Age1**
Stud#1	**AuthorPublication**	**Paper1**
Paper#1	**Type**	**Publication**
...	...	...

FIGURE 1.5 – Schéma des données selon l'approche verticale

implique plusieurs auto-jointures. Plusieurs systèmes de BDBO utilisent cette représentation, tels que *RStar* [128], *RDFSuite* [9], *KAON* [208] et *Sesame* [28]. Une étude comparative entre ces différents systèmes est présentée dans [1].

Une variante de cette approche appelée approche *verticale avec ID* consiste à stocker dans une autre table la correspondance entre les URI des ressources et un identifiant ID, afin de représenter dans la table de triplets uniquement l'identifiant ID (dans le but de limiter le coût de stockage et faciliter l'accès aux données).

En prenant l'exemple du schéma d'ontologie décrit en figure 1.4, la représentation verticale de cette ontologie donnerait la table représentée en figure 1.5.

Dans une approche **binaire**, les classes et les propriétés auxquelles appartienent les instances ontologiques sont stockées dans des tables de structures différentes. *RDFSuite* [9], *Genea* [112], *IBM SOR* [124] et *DLDB-OWL* [148] sont des exemples de BDBO qui utilisent cette représentation. *RDFSuite* supportent les deux représentations verticale et binaire. Cette approche n'est généralement pas adaptée aux larges ontologies et souffre de l'altération du schéma lorsque l'ontologie évolue. Cette approche a trois variantes : table unique, ISA et NOISA.

– La *variante table unique* utilise une seule table pour stocker toutes les classes de l'ontologie et une table pour chaque propriété. La table des classes contient deux colonnes : *uri* pour stocker l'identifiant des instances et une colonne *classID* pour représenter l'identifiant de la classe.

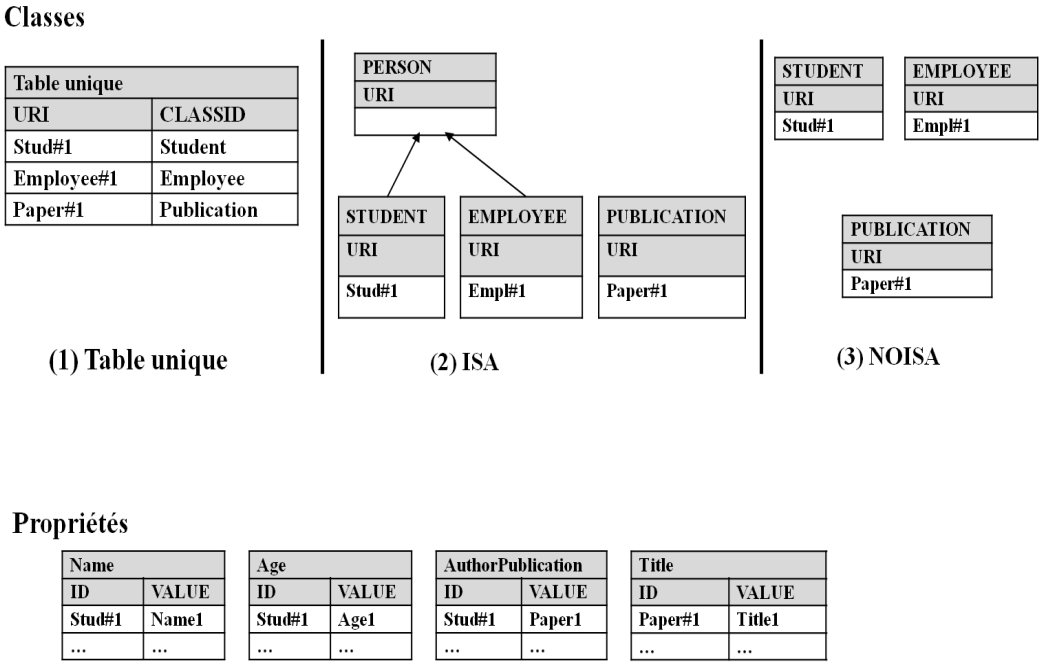

FIGURE 1.6 – Schéma des données selon l'approche binaire

- La *variante ISA* consiste à utiliser les caractéristiques des SGBD relationnels objets pour représenter l'héritage des classes et des propriétés en utilisant la définition sub-table. Chaque table d'une classe contient une seule colonne représentant les identifiants des instances de cette classe.
- La *variante NOISA* consiste à ne pas utiliser l'héritage des tables. Les tables de classes et de propriétés sont définies séparément sans être mises en relation.

Dans les trois variantes, la table de chaque propriété est constituée de deux colonnes : id pour l'identifiant de la propriété, et value pour la valeur de la propriété. En prenant l'exemple du schéma d'ontologie décrit en figure 1.4, la représentation binaire de cette ontologie donnerait les tables représentées en figure 1.6.

L'approche **horizontale** consiste à associer à chaque classe de l'ontologie une table d'instances ayant une colonne pour chaque propriété associée à une valeur pour au moins une instance de la classe. Les BDBO *OntoDB* [87], *OntoDB2* [56] et *OntoMS* [149] utilisent cette représentation. L'héritage des tables est représenté si le SGBD utilisé le permet. Ce schéma de stockage est similaire à celui utilisé dans les BD conventionnelles. En prenant l'exemple du schéma d'ontologie décrit en figure 1.4, la représentation horizontale de cette ontologie donnerait les tables représentées en figure 1.7.

Quelques systèmes utilisent une approche *hybride* comme : *Jena2* [212] qui combine la représentation verticale et binaire (en représentant les tables triplets et en regroupant quelques attributs) ou *DLDB* [148] qui combine l'approche binaire et horizontale. Ces différentes représentations des DBO sont utilisées selon les besoins de l'application à mettre en œuvre. Des tests réalisés sur un ensemble de requêtes montrent que les performances des BDBO dépendent de plusieurs paramètres (requêtes utilisées, ontologie, type du SGBD utilisé). Nous citons quelques

Student			
ID	Name	Age	AuthorPubli
Stud#1	Name1	Age1	Paper#1
...	...		

Employee		
ID	Name	Age
Emp#1	Name1	35
...	...	

Publication	
ID	Title
Paper#1	Title#1
...	...

FIGURE 1.7 – Schéma des données selon l'approche horizontale

exemples [87, 136, 56] :

- La représentation horizontale est plus adaptée lorsque les instances ontologiques sont décrites par un grand nombre de propriétés communes et si les requêtes utilisées portent sur un grand nombre de propriétés (à partir de 6).
- L'approche binaire est plus appropriée lorsque les instances d'une classe sont décrites par des propriétés différentes. Le coût de stockage est plus réduit par rapport à une représentation horizontale qui aurait nécessité plusieurs champs nuls. La mise à jour des données est cependant plus coûteuse car cela nécessite des traitements sur plusieurs tables. L'utilisation d'un SGBD C-Store peut également améliorer les performances des requêtes sur une représentation binaire [3].

1.2.2.4 Architectures des BDBO

Une BDBO peut utiliser un seul ou plusieurs schémas de données pour stocker l'ensemble de ses données, ce qui a donné lieu à trois principales architectures cibles [56] : architecture de *type I*, *type II* et *type III*. Le groupe OMG (Object Management Group) définit une architecture de conception organisée en quatre niveaux d'abstraction : les données du monde réel (M0), le modèle (M1), le métamodèle (M2) et le méta métamodèle (M3). La figure 1.8 montre l'évolution des architectures de BDBO selon ces quatre niveaux d'abstraction.

Dans les BDBO de **type I**, comme dans les BD classiques, l'ontologie et les données (DBO) sont stockées dans un même schéma (niveau M0 et M1). Il n'y pas de séparation entre l'ontologie et les données. Cette architecture utilise généralement la représentation verticale où toutes les informations de l'ontologie sont représentées en utilisant un seul schéma composé d'une seule table de triplets ou une table verticale. Cette architecture permet d'effectuer facilement la mise à jour des instances et des propriétés (comme les applications de e-commerce qui nécessitent plusieurs mises à jour). Ses performances sont cependant dégradées à cause des nombreuses opérations d'auto-jointure effectuées sur une table unique. *Jena2* [212] utilise ce type d'architecture.

Dans une architecture de **type II**, l'ontologie et les données associées sont stockées dans deux schémas différents : un schéma pour les DBO, et un autre pour l'ontologie. Ce dernier est spécifique au formalisme d'ontologie supporté. Les performances de cette architecture sont meilleures que celles d'une architecture de type I. Elle présente cependant certaines limites : (i) le schéma de l'ontologie est figé et (ii) l'introduction de concepts issus d'autres modèles d'ontologies n'est pas permise. Les systèmes *RDF Suite* [9] et *SOR* [124] utilisent cette architecture.

L'architecture de **type III** étend celle de type II en définissant un schéma supplémentaire appelé *métaschéma* (niveau M2 et M3). L'ontologie constitue ainsi une instance de ce métaschéma. La présence du métaschéma offre une flexibilité pour l'ontologie et permet : (i) l'évolution du

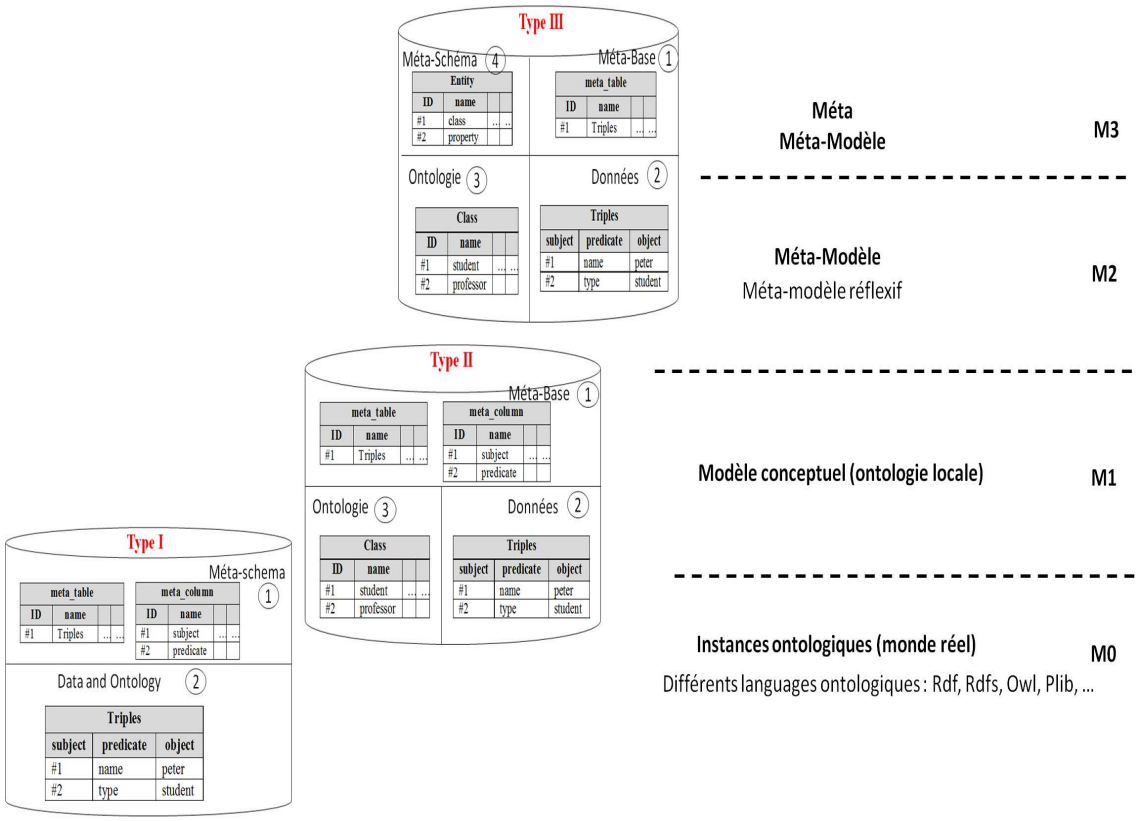

FIGURE 1.8 – Trois architectures de BDBO

formalisme ontologique utilisé, (ii) un accès générique aux ontologies ainsi qu'aux données et (iii) le stockage de différents constructeurs de formalismes d'ontologies (OWL, DAML+OIL, PLIB, etc). Les systèmes *OntoDB* [87] et son extension *OntoDB2* [56], développés au laboratoire LIAS présentent des BDBO de cette architecture.

Après avoir étudié le rôle des ontologies dans la conception des BD, nous analysons dans ce qui suit leur utilisation dans l'ingénierie des besoins (IB). Nous étudions ensuite l'impact de la persistance des besoins sur le cycle de vie de l'ED.

1.2.3 Les ontologies au service de l'ingénierie des besoins

Le processus d'IB inclue des activités de découverte, de spécification, de négociation et de validation des besoins. Les besoins à l'égard du système sont répertoriés dans un document de spécification des besoins [159].

De nombreuses enquêtes confirment l'importance de l'IB dans la réussite des projets de systèmes d'information. Une enquête menée auprès de 800 projets conduits dans 350 compagnies américaines par le Standish Group [191] et présentée dans deux rapports, intitulés 'Chaos' et 'Unfinished Voyages', a révélé que 31% des projets sont annulés avant même d'être terminés, et que 50% des projets n'avaient que partiellement réussi car ils avaient nécessité des budgets et des délais très fortement majorés [159]. La mauvaise qualité des documents de besoins constitue 47% des causes d'échecs citées. Ce pourcentage est distribué de la façon suivante [159] : manque de

participation des utilisateurs (13%), besoins mal exprimés (ou incomplets) (12%), besoins changés entre le début et la fin du projet (11%), besoins qui manquent de réalisme (6%), et objectifs peu clairs (5%). De ces constats, il a été nécessaire de définir des méthodes, des techniques et des outils qui permettent d'analyser, de valider et de représenter de manière adéquate et structurée les besoins relatifs au système à développer.

Nous analysons dans ce qui suit l'intérêt de l'utilisation des ontologies pour la définition des besoins, et le rôle des ontologies pour résoudre les problèmes relevés.

1.2.3.1 L'utilisation des ontologies et l'ingénierie des besoins

Plusieurs travaux ont utilisé les ontologies en IB afin de fournir une représentation adéquate des besoins. La fusion entre ces deux domaines (ingénierie des besoins et ontologies) a suscité l'intérêt de la communauté de recherche depuis les années 80 [11] et encore dans travaux récents, où les ontologies ont montré leur efficacité pour la spécification des besoins, leur unification, leur formalisation et le raisonnement sur les besoins [34]. Nous citons quelques projets initiateurs ayant utilisé les connaissances ontologiques dans la définition des besoins [77, 139, 52].

Le projet *RML* (Requirements Modelling Language) a donné lieu à de nombreuses études alliant les ontologies à l'IB. RML est un langage de modélisation des besoins proposé dans le début des années 80. RML a été développé selon les principes suivants [77] : (1) l'écriture des besoins et leurs spécifications fonctionnelles doivent être fournies. (2) Les besoins doivent être élaborés et présentés sous forme de modèles. (3) Les efforts d'abstraction et de raffinement sont significatifs dans l'IB. (4) Les langages de modélisation formelle des besoins sont nécessaires. RML défini son propre langage ontologique. En utilisant ce langage d'ontologie, RML fournit une ontologie pour la modélisation des besoins.

Telos [139] fournit un Framework de modélisation des besoins en se basant sur les travaux de RML. Le Framework vise à étendre le rôle de l'ontologie en permettant l'inclusion des concepts d'agents, d'objectifs, d'acteurs et de tâches, au modèle de besoins. D'autres améliorations ont été réalisées par Telos comme l'inclusion des intervalles de temps et des relations temporelles permettant de garder l'historique du domaine représenté.

Le Framework *Kaos* [52] fournit un langage de modélisation des besoins, basé sur la logique du premier ordre temporel. KAOS offre plusieurs formes de représentations des besoins utilisant : une ontologie générique qui fournit un méta-modèle pour les besoins, et une couche conceptuelle.

Les ontologies sont utiles pour la représentation de nombreux types de connaissances. La nature de l'IB consiste à capturer les connaissances provenant de plusieurs sources. Il y a donc plusieurs utilisations possibles des ontologies dans l'ingénierie des besoins. Les ontologies ont été utilisées pour la définition des besoins pour leurs principales caractéristiques que nous allons détailler dans ce qui suit [34] : la *spécification*, le *raisonnement* et la *modularité ontologique*.

1.2.3.1.1 Spécification

De manière informelle, l'ontologie permet d'assister le concepteur durant la phase d'identification des besoins dans la compréhension des relations entre les différents composants d'un domaine. Par ailleurs, l'ontologie fournit une représentation formelle offrant des mécanismes suffisamment expressifs pour représenter le modèle des besoins. La norme IEEE 830-1998 [5] pour

5. http ://www.math.uaa.alaska.edu/∼afkjm/cs401/IEEE830.pdf

la spécification des besoins a défini huit principales caractéristiques que toute spécification des besoins doit assurer. Une spécification doit être : (1) correcte, (2) sans ambiguïté, (3) complète, (4) consistante, (5) stable, (6) vérifiable, (7) modifiable et (8) traçable. Nous détaillons ces caractéristiques.

(1) Spécification correcte : une spécification des besoins doit être comparée à toute spécification de niveau supérieur (comme la spécification des besoins de l'ensemble du système), ainsi qu'avec d'autres normes applicables afin de s'assurer que la spécification est correcte. Alternativement, le client ou l'utilisateur peut déterminer si la spécification reflète correctement leurs besoins réels.

(2) Spécification sans ambiguïté : une spécification est sans ambiguïté si, et seulement si, toutes les exigences qui y sont énoncées ne possèdent qu'une seule interprétation. Chaque caractéristique du produit final doit être décrite en utilisant un seul terme unique. Dans les cas où un terme utilisé dans un contexte particulier pourrait avoir des significations multiples, le terme devrait être inclus dans un glossaire où son sens est spécifié.

(3) Spécification complète : une spécification est complète si, et seulement si, elle vérifie les critères suivants :

- Toutes les exigences importantes sont exprimées, qu'il s'agisse de fonctionnalités, de performances, de contraintes de conception, d'attributs, ou d'interfaces externes.
- La spécification permet la définition des réponses du logiciel pour toutes les données d'entrée.
- La spécification comprend une définition de tous les labels, références, figures, tableaux et illustrations qui y figurent, ainsi que la définition de tous les termes et unités de mesure utilisées.

(4) Spécification consistante ou cohérente : une spécification est cohérente si, et seulement si, aucun sous-ensemble des exigences individuelles décrites ne présente un conflit. Ces conflits peuvent être de trois types :

- Les caractéristiques spécifiées des objets sont en conflits : se produit lorsque par exemple, le format de sortie d'un besoin dans un premier rapport est différent de la spécification du même besoin dans un autre rapport.
- Conflits temporels ou logiques : se produit lorsque par exemple, un besoin présente une seule entrée d'un programme alors qu'un autre besoin présente plusieurs entrées.
- Deux besoins peuvent spécifier le même concept avec des termes différents.

(5) Spécification Stable : une spécification est dite stable et triée (classée) par importance si chaque besoin a un identifiant qui indique son rang d'importance ou sa stabilité.

(6) Spécification vérifiable : une spécification est vérifiable si, et seulement si, toutes les exigences qui y sont énoncées sont vérifiables. Une exigence est vérifiable si, et seulement s'il existe un procédé avec lequel une personne ou une machine peut vérifier que le produit logiciel répond à cette exigence. En général, toute exigence ambiguë et non quantifiable n'est pas vérifiable.

(7) Spécification modifiable : une spécification est modifiable si, et seulement si, sa structure et son style sont construits de telle sorte que, toute modification des exigences peut être facilement et complètement mise en œuvre. Cette modification ne doit pas altérer la structure et le style de la spécification.

(8) Spécification traçable : une spécification est traçable si chacune de ses exigences est claire et si la spécification facilite le référencement de chaque exigence dans le développement du système ou dans sa documentation.

Selon ces critères établis par la norme IEEE 830-1998, l'utilisation d'une ontologie de domaine dans la phase de spécification des besoins contribue à assurer une spécification des besoins *correcte*, *complète*, *vérifiable* et *stable*. L'ontologie présente le principal avantage de permettre la réutilisation du savoir sur un domaine et expliciter ce qui est considéré comme implicite sur un domaine. L'utilisation des ontologies permet de mieux cerner les limites des connaissances nécessaires pour la modélisation du système et contribue ainsi à la *complétude* de la spécification des besoins. L'aspect formel des ontologies permet de modéliser explicitement les besoins des utilisateurs d'une manière interprétable par une machine. Ceci contribue à fournir une spécification des besoins *vérifiable* et donc *correcte*. Les ontologies présentent des modèles consensuels d'un domaine, établis par un consensus d'experts. Ces connaissances ne sont donc pas sujettes à des modifications majeures. Le maintien et l'évolution des besoins s'en trouvent par conséquent limités. Cette caractéristique contribue à la *stabilité* de la spécification des besoins. L'ontologie permet d'assurer au concepteur de couvrir totalement les deux caractéristiques de *non ambiguïté* et de *cohérence* des besoins. L'ontologie assure la *non-ambiguïté* de la spécification des besoins, dans le sens où elle fournit une seule interprétation unique pour chaque besoin. Une ontologie définit un vocabulaire commun pour les utilisateurs qui doivent partager l'information sur un domaine particulier. Projeter les besoins des utilisateurs sur une ontologie, permet ainsi d'unifier leurs besoins, d'éliminer les redondances et de résoudre les conflits sémantiques issus des différents points de vue, idées et propositions des utilisateurs. L'ontologie assure la *cohérence* de la spécification des besoins dans le sens où elle permet d'éliminer les relations conflictuelles entre les besoins.

Même si nous n'abordons pas la problématique de traçabilité des besoins dans notre étude, nous notons que la persistance des besoins contribue à assurer la dernière caractéristique de la norme IEEE, à savoir la *traçabilité* des besoins.

1.2.3.1.2 Raisonnement

Les besoins exprimés par les utilisateurs sont souvent complexes et peuvent même être contradictoires car ils sont issus de différents utilisateurs. La projection du modèle des besoins sur une ontologie permet de détecter ces incompatibilités en utilisant les mécanismes de raisonnement nécessaires pour vérifier la consistance du modèle des besoins. Ces mécanismes sont actuellement supportés et automatisés par différents raisonneurs comme *Fact*, *Racer* ou *Pellet*.

1.2.3.1.3 Modularité ontologique

Dans les projets qui font intervenir plusieurs domaines, la modularité des ontologies permet d'extraire les modules (fragments) nécessaires à partir de différentes ontologies de domaine pouvant être utilisées comme support au modèle des besoins. La définition d'un module d'une ontologie est un aspect important dans notre méthode. En effet, les ontologies de domaine *réelles* modélisent différents aspects d'un domaine à plusieurs niveaux de granularité [95]. Pouvoir définir un module de cette ontologie est essentiel pour différentes raisons :

– Les concepteurs d'une application d'entreposage ne sont pas forcément experts du domaine.

Ils peuvent ainsi *emprunter* la connaissance des concepts à partir d'ontologies externes consensuelles et y extraire les modules qui répondent à leurs besoins.
- Dans les projets faisant intervenir plusieurs domaines, il est important de pouvoir définir une ontologie décrivant l'application à développer à partir de plusieurs modules extraits de différentes ontologies de domaine.
- Il a été démontré que les raisonneurs actuels ne sont efficaces que sur les ontologies de petite taille [216]. Définir des modules d'ontologie, représentant le noyau sémantique de l'application à développer, permet ainsi de faciliter les raisonnements.

La définition d'un module d'ontologie se fait généralement par la spécification d'une *signature* (ensemble de termes à extraire). Le module extrait correspondra aux termes spécifiés ainsi qu'à l'ensemble des axiomes pertinents pour ces termes. La signature est représentée dans notre approche par les termes du dictionnaire de données. Différentes méthodes ont été proposées permettant d'extraire des modules d'ontologie comme les logiques de description distribuées [26], E-connections [51], la segmentation d'ontologies [170], OntoPath [94], etc. *Wang et al.* [216] présentent un état de l'art sur les méthodes et formalismes de modularité d'ontologies proposés dans la littérature. Nous avons opté dans notre approche, pour la méthode définie dans [75] que nous présenterons dans la section suivante (cf. section 1.3.4.2.2).

Après avoir étudié l'intérêt de l'utilisation des ontologies pour la définition des besoins, nous montrons dans ce suit l'impact de la persistance des besoins au sein de la structure de l'ED.

1.2.4 Impact de la persistance des besoins sur le cycle de vie de l'ED

Les méthodes de conception d'ED orientées besoins ou mixtes reconnaissent la nécessité d'identifier et d'analyser les besoins des utilisateurs dans le processus de conception, mais aucune de ces méthodes n'analyse leur importance au-delà des étapes de modélisation conceptuelle et/ou logique.

Pour réaliser un produit d'ED autour du cycle de vie cité, nous avons identifié l'intervention de plusieurs acteurs appartenant à différents corps de métiers : le *rédacteur* du cahier des charges en relation avec le client (souvent membre de l'organisation qui souhaite fabriquer l'ED). Ce dernier a toujours besoin des *utilisateurs futurs* du produit final afin d'identifier les besoins, les contraintes, etc. Le *concepteur* a pour mission de concevoir le schéma de l'ED selon un langage de modélisation (E/A, UML, etc.). Le modèle logique de l'ED est généré à partir du modèle conceptuel selon le choix de la solution de persistance souvent négociée par le *client* avec les *représentants des éditeurs de SGBD*. L'ED est accédé par des utilisateurs finaux qui peuvent être ceux impliqués dans la phase de collecte des besoins. Afin de satisfaire les exigences de ces utilisateurs (temps de réponse de traitement, sécurité, etc.), l'*administrateur* réagit au niveau de la phase de conception physique. Par exemple, pour optimiser les accès à l'ED, il peut créer des techniques d'optimisation comme les vues matérialisées, le partitionnement, les index, la compression, etc. La figure 1.9 recense les principaux acteurs et les phases du cycle vie dans lesquelles ces acteurs interviennent [108]. Nous avons constaté lors de notre analyse du cycle de conception des BD/ED, la multiplicité des modèles de stockage et des architectures supportant ces systèmes de gestion de données. Cette diversification offre plusieurs choix de conception et exige une communication étroite entre les différentes phases du cycle de vie. Par exemple, les négociateurs du client et les représentants des éditeurs doivent répondre aux choix pertinents des modèles et des architectures. Ces choix

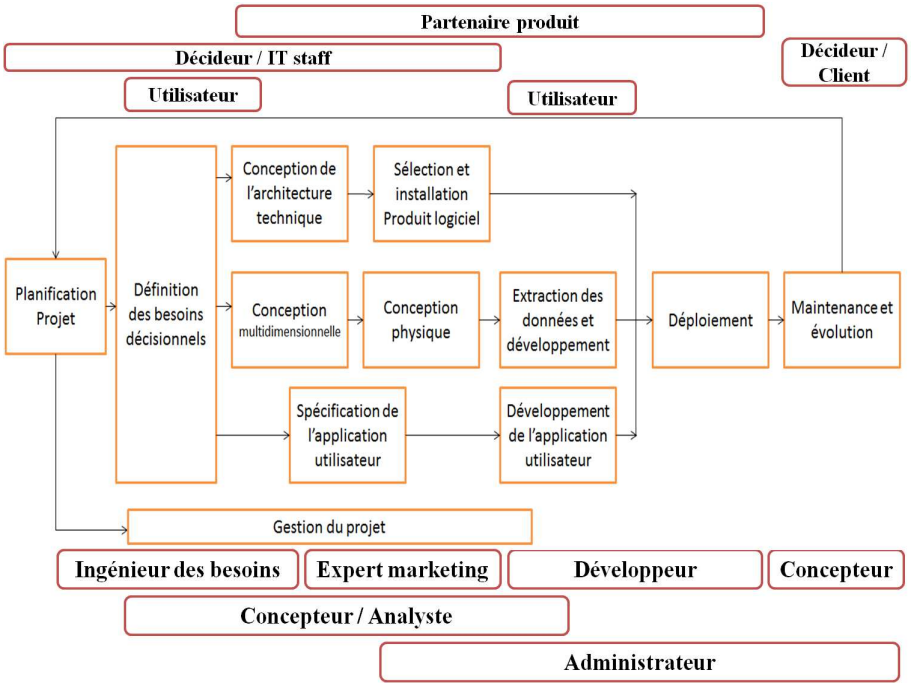

FIGURE 1.9 – Acteurs intervenant dans les phases du cycle vie d'un projet décisionnel

dépendent des données et des traitements. Si l'entreprise possède un administrateur, ce dernier à l'aide de simulations (avec des bancs d'essai) peut donner des recommandations pour acquérir la technologie susceptible de gérer l'ED. Pour pouvoir effectuer de telles simulations, l'unique ressource disponible consiste en l'ensemble des besoins des utilisateurs qui permettent de fournir le modèle de données (l'ensemble des entités et propriétés) et aussi les traitements (requêtes et fonctions) de l'ED. Ces besoins sont exploités à différentes phases du cycle de vie de l'ED ; leur représentation au sein de l'ED est donc nécessaire. Nous discutons dans cette section l'importance de la persistance des besoins dans la structure finale de l'ED en montrant le rôle des besoins pour plusieurs phases du cycle de vie de l'ED.

1.2.4.1 La phase de conception

L'exploitation des besoins des utilisateurs permet dans un premier temps de générer le modèle conceptuel multidimensionnel de l'ED répondant aux exigences des décideurs et utilisateurs. L'exploitation des besoins se fait également sentir lors du choix du schéma de stockage logique. Ce schéma peut suivre différentes représentations : relationnelle, multidimensionnelle, hybride. D'autres schémas de stockage d'ED sont discutés dans [74]. Le choix de la structure de représentation logique la plus appropriée dépend fortement des besoins de l'application et des requêtes des utilisateurs. Par exemple, les modèles relationnels sont utilisés lorsque les requêtes sont très complexes nécessitant plusieurs opérations coûteuses comme les jointures. Les modèles de stockage multidimensionnels sont utilisés pour favoriser l'accès rapide aux données.

L'utilisation des besoins se fait également ressentir à la phase de modélisation physique pour différentes tâches d'optimisation. Certains besoins des utilisateurs définissent les traitements auxquels devra répondre l'ED. Ces besoins, qui se traduisent en requêtes, peuvent être utilisés

pour définir certaines structures d'optimisation. Plusieurs SGBD actuels (comme Oracle ou SQL Server) stockent leurs requêtes afin de les exploiter plus tard lors du processus d'optimisation. D'autres travaux reposent principalement sur les caractéristiques de la charge de requêtes exprimées par les utilisateurs pour définir des structures d'optimisation de l'ED [110]. Il se trouve cependant que cette charge de requêtes n'est constituée qu'après une certaine période d'exploitation de l'ED suite à son développement. Ces requêtes sont exprimées sous forme de besoins par les utilisateurs dès le début du projet, et peuvent ainsi être exploitées pour prévoir des structures d'optimisation. Nous avons mené une telle étude, que nous présentons dans la section (cf. section 1.4), où nous identifions différentes structures d'optimisation de l'ED à partir d'un ensemble de besoins. Le stockage des besoins dans l'ED leur fournit une meilleure représentation et permet aussi leur restitution à tout moment de la conception.

1.2.4.2 La phase d'exploitation

L'ED une fois conçu, est exploité par les décideurs pour différentes tâches d'analyse afin de faciliter le processus de prise de décision. Nous discutons ici l'importance des besoins pour la phase d'exploitation principalement pour les tâches de personnalisation et recommandation. La richesse des ED et la masse de données qu'ils stockent ont influencé leur exploitation par les utilisateurs qui ont des préférences et des profils qui sont souvent liés au contenu de l'ED. Ces préférences sont gérées lors de la phase de personnalisation. Elles représentent un concept-clé permettant de filtrer le flux d'informations. Les préférences sont considérées comme des besoins non fonctionnels, et peuvent donc être identifiées dès la phase de collecte de besoins. Plusieurs travaux portant sur la personnalisation et recommandation des requêtes comme [17] et [71] supposent l'existence et la connaissance des préférences des utilisateurs pour évaluer les requêtes, ou demandent leur explicitation durant la phase de personnalisation pour chaque requête émise. Mais comme mentionné dans [71], il est préférable de libérer l'utilisateur de cette tâche. La persistance du modèle de besoins permet le stockage des préférences des utilisateurs, répondant ainsi au problème soulevé.

1.2.4.3 La phase de maintenance

Comme dans tout projet logiciel, la maintenance d'un ED est une étape nécessaire qui assure sa bonne qualité. La gestion de la qualité en développement logiciel commence par la compréhension des besoins des utilisateurs [11]. La qualité de tout système se mesure en fonction de sa capacité à répondre aux buts et objectifs fixés. Par exemple, *Vassiliadis et al.* [200] proposent une méthode de gestion de la qualité de l'ED basée sur le paradigme Goal/Question/Metric (GQM) [199], où la qualité de l'ED est évaluée en fonction des buts émis. Ces méthodes supposent la disponibilité des buts des utilisateurs. L'évolution d'un ED repose également sur l'identification des différents changements qui surviennent. La traçabilité des besoins permet la détection de ces changements et nécessite ainsi leur connaissance dès le début du projet. La traçabilité des besoins est aussi utilisée dans le processus de réutilisation des composants [11]. Lorsque des similarités entre les besoins sont identifiées, cette information est utilisée pour identifier les composants réutilisables. Par conséquent, le stockage des besoins dans la structure de l'ED permet d'assurer la traçabilité des besoins depuis leur identification, améliorant ainsi le processus de maintenance

et d'évolution de l'ED.

Après avoir étudié les fondements théoriques nécessaires à l'établissement de notre proposition, nous présentons dans ce qui suit la méthode de conception du schéma de l'ED que nous proposons.

1.3 Définition du système d'ED manipulant données et traitements

Notre objectif à travers cette étude est de définir un mécanisme de persistance des besoins des utilisateurs au sein d'un ED sémantique. Pour atteindre cet objectif, nous devons d'abord fournir un modèle de besoins et une formalisation du modèle de l'ontologie. Nous commençons cette section par présenter les hypothèses que nous posons pour l'ontologie de domaine. Nous décrivons ensuite l'étude de cas que nous utilisons pour illustrer notre approche. Nous présentons les deux entrées de notre méthode (le modèle de l'ontologie et le modèle des besoins), et nous exposons enfin le processus de conception permettant d'obtenir à un ED sémantique faisant persister les besoins.

1.3.1 Hypothèses

Pour réaliser nos propositions, nous définissons deux principales hypothèses concernant l'ontologie de domaine.

1.3.1.1 Hypothèse 1

L'ontologie est utilisée dans notre méthode pour faciliter l'intégration des sources et la spécification des besoins. Les ontologies ont longuement été utilisées pour assurer une intégration automatique des données hétérogènes. Cette intégration suppose l'existence d'une ontologie de domaine, et de sources ontologiques référençant cette ontologie [16]. Une ontologie est dite "partagée" entre plusieurs sources, lorsque les sources s'engagent sur les concepts qu'elle définit et sur le fait d'utiliser les définitions ontologiques de l'ontologie partagée qui ont été acceptées et éventuellement normalisées. Afin de garder son autonomie, chaque source peut définir sa propre hiérarchie de classes, et, si besoin est, rajouter les propriétés qui n'existent pas dans l'ontologie partagée [16]. Nous considérons dans la présente étude que l'ontologie partagée est existante et disponible.

1.3.1.2 Hypothèse 2

L'ontologie de domaine est supposée remplir les hypothèses de *typage fort des propriétés* et de *complétude de définition*. Ces contraintes sont généralement imposées lors du stockage des ontologies dans une BD comme dans [87, 56, 9] :

1.3.1.2.1 Typage fort des propriétés

– Toutes les propriétés des classes doivent avoir obligatoirement un domaine et un co-domaine. Cette exigence provient du fait qu'en langage OWL et RDF Schéma, une propriété peut exister sans domaine et/ou co-domaine. Lorsque le co-domaine d'une propriété n'est pas défini, par défaut, elle aura pour co-domaine l'union des classes et des types littéraux. Cette union est dépourvue de sens. De plus il y a des risques d'ambiguïté lors de l'héritage de propriétés [9].
– Le domaine et le co-domaine d'une propriété doivent être uniques. Cette exigence vient toujours du fait qu'en OWL et RDF Schema, il est possible de déclarer plusieurs domaines et co-domaines d'une même propriété. Sans cette exigence, il y aurait une ambiguïté dans le choix du type des colonnes des tables des propriétés.

1.3.1.2.2 Complétude de définition

Les descriptions complètes de tous les concepts qui contribuent à la définition d'un concept doivent pouvoir être représentées dans le même environnement (exemple : fichier) que celui où ce concept est défini. Ceci suppose en particulier que les superclasses d'une classe, si elles existent, soient connues, de même que les domaines et le co-domaine de ses propriétés.

Nous présentons dans ce qui suit l'étude de cas que nous utilisons pour illustrer la méthode proposée.

1.3.2 Etude de cas

L'étude de cas que nous utilisons repose sur la spécification du banc d'essai *Star Schema Benchmark*[6] (SSB). La spécification du banc d'essai SSB fournit un schéma logique de données (relationnel) d'un ED, un ensemble de besoins décisionnels et un ensemble de requêtes SQL défini à partir des besoins. La figure 1.10 présente le modèle conceptuel de la spécification du banc d'essai, obtenu par un processus de rétro-conception appliqué sur le schéma normalisé du modèle logique de SSB. Nous avons conçu une ontologie de domaine à partir de ce modèle conceptuel. Nous considérons ce modèle comme le modèle de l'ontologie partagée. Ce modèle est représenté par un diagramme de classes. Il comporte différentes classes relatives au domaine des transactions commerciales. La classe centrale est la classe *Order*, liée aux classes *Customer*, *Supplier*, *Part-product* et *Time*. D'autres classes explicitent la localisation du client comme *City*, *Nation* et *Region*. Les classes *Category* et *Brand* explicite le type et la marque du produit.

Le choix d'utiliser la spécification du banc d'essai SSB s'explique par deux raisons :

– SSB est dédié aux applications décisionnelles de type ED,
– SSB fournit le schéma de données, les besoins décisionnels et les requêtes des utilisateurs.

Les besoins et les requêtes des utilisateurs sont des entrées importantes pour notre méthode, dans la mesure où nous validerons notre méthode en montrant l'impact de la persistance des besoins pour simuler certaines tâches du cycle de vie de l'ED, principalement la tâche d'optimisation. Nous comparons dans nos expérimentations le coût d'une optimisation dirigée par les besoins (obtenus dès le début de la conception de l'ED), et une optimisation dirigée par les requêtes des utilisateurs (obtenues après la conception de l'ED et son exploitation par les utilisateurs). Cette comparaison nous permet d'une part de faciliter la tâche de l'administrateur/concepteur,

6. http ://www.cs.umb.edu/ poneil/StarSchemaB.PDF

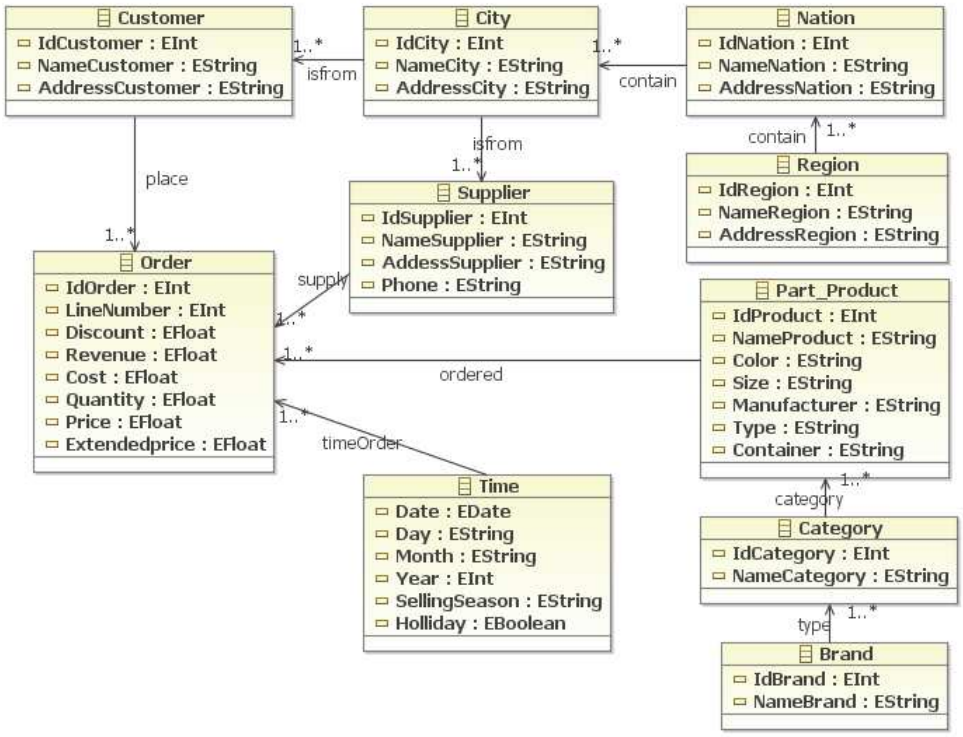

FIGURE 1.10 – Ontologie du schéma du banc d'essai SSB

en prédisant certaines structures d'optimisation pertinentes. D'autre part, cette comparaison nous permet de montrer si une simulation des phases de conception de l'ED est possible à partir des besoins des utilisateurs. Le banc d'essai SSB, certes fournit un schéma de données logique (et pas conceptuel comme le préconise notre méthode), mais ce schéma relationnel est bien formé et, pour cette raison, il a été cohérent de déduire une ontologie OWL à partir de ce schéma, par un processus de rétro-conception.

Nous considérons le schéma de données SSB comme l'ontologie partagée par les sources de données. Nous considérons également les besoins décisionnels (*business questions*) définis dans la spécification du banc d'essai SSB comme les besoins exprimés par les utilisateurs dans notre méthode.

1.3.3 Formalisation des entrées de la méthode

Le processus de conception que nous proposons repose sur deux entrées principales : (1) un ensemble de besoins des utilisateurs exprimés sous forme de buts et (2) une ontologie de domaine partagée par un ensemble de sources de données. Nous formalisons dans ce qui suit ces deux entrées.

1.3.3.1 Le modèle des besoins des utilisateurs

Il y a un certain nombre de possibilités d'évaluer et d'analyser un ensemble de besoins en cours d'élaboration. La construction de modèles est l'une des méthodes les plus utilisées et reconnues

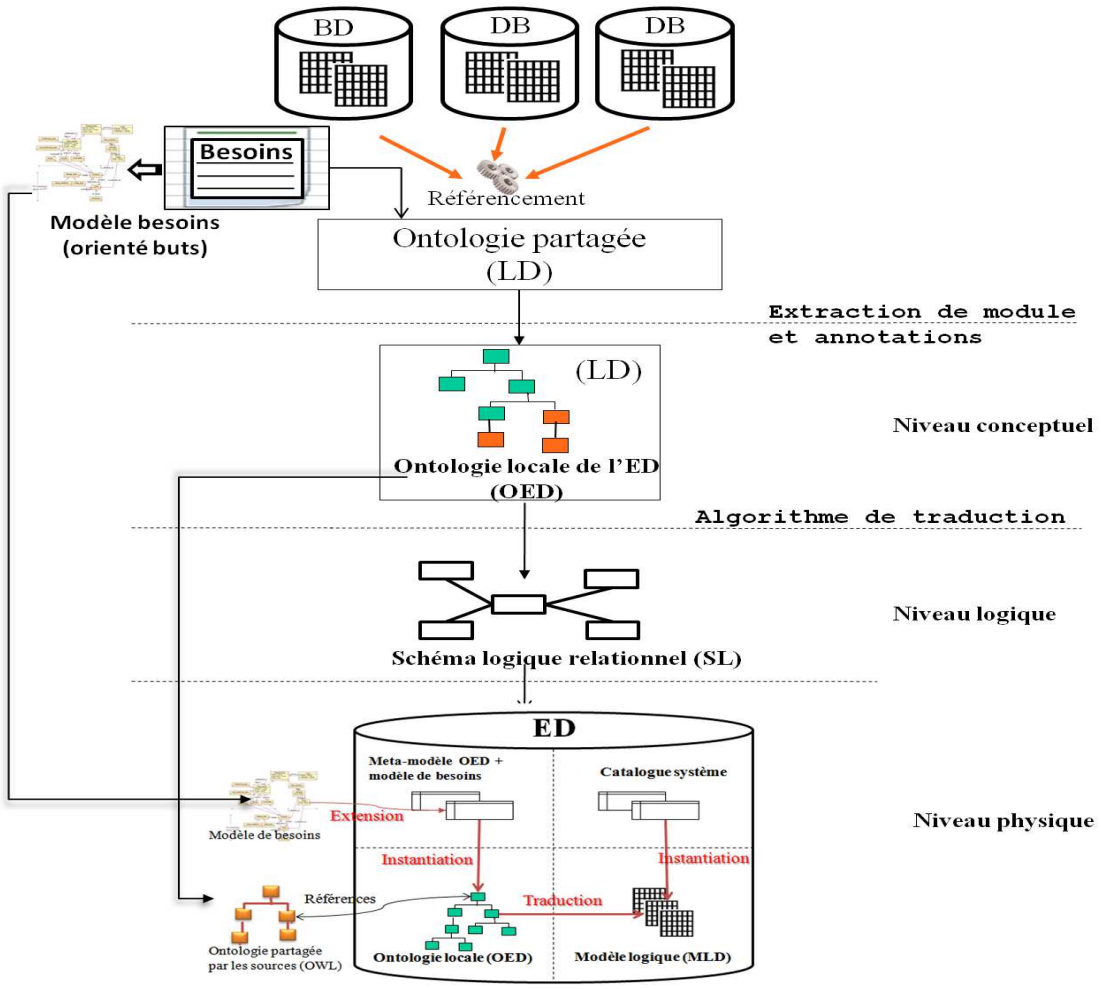

FIGURE 1.11 – Approche globale de conception du schéma l'ED

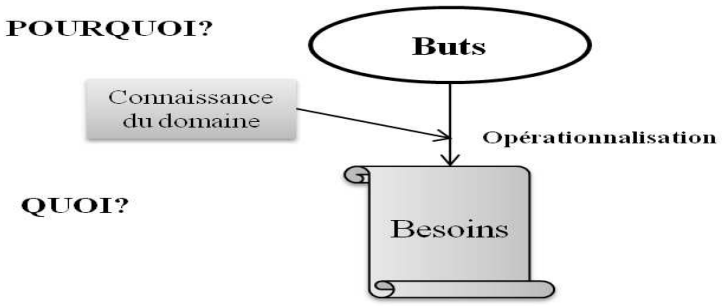

FIGURE 1.12 – Le fondement de l'ingénierie des besoins [159]

dans l'ingénierie des besoins. Fournir un modèle des besoins permet de tester la faisabilité des besoins et aide à comprendre les aspects du domaine étudié. Nous avons opté dans notre méthode de conception pour un modèle de besoins *dirigé par les buts*. Nous avons proposé un modèle orienté but que nous avons défini comme un modèle ***pivot*** issus de trois approches dirigées par les buts, fréquemment utilisées dans la spécification des besoins dans les projets d'entreposage, qui sont : *KAOS*, *IStar* et *Tropos*.

Nous commençons par décrire brièvement les caractéristiques des approches dirigées par les buts. Nous procédons ensuite à la présentation des trois approches citées, pour définir le modèle pivot.

1.3.3.1.1 Les approches d'IB dirigées par les buts

Le rôle de l'IB est de déterminer les fonctionnalités que le système doit mettre en œuvre pour aider à la satisfaction des buts organisationnels ; et d'identifier les contraintes qui restreignent la mise en œuvre des fonctions du système [159]. Les approches conventionnelles de l'IB centrées sur la question "quoi" formalisent les fonctionnalités imposées au système. L'IB ne se limite plus à prendre en compte ce que doit faire le système mais cherche à comprendre le "pourquoi" du système en termes d'objectifs organisationnels et d'expliquer l'impact de ces objectifs sur le système de l'organisation. Comme illustré dans la figure 1.12, *Axel van Lamsweerde* [197] schématise cette vue de l'IB centrée sur les questions "pourquoi" et "quoi" ainsi que l'association d'un "quoi" motivé par le "pourquoi" [159]. Le rôle des approches orientées but est de répondre au "pourquoi". Un but se définit comme un objectif que le futur système doit garantir. *Kaizhi Yue* [218] a permis l'introduction des modèles de buts dans le document de spécification des besoins afin de fournir un critère de complétude des besoins : une spécification des besoins est *complète* si elle permet de satisfaire le but qu'elle affine [159]. Les approches dirigées par les buts proposent des mécanismes logiques pour l'identification, l'organisation et la justification des exigences d'un système. Les approches orientées buts ont rapidement été adoptées parmi les approches d'IB où elles ont présenté de nombreuses contributions [52, 218, 159] :

- Ces approches permettent de déterminer les objectifs du système à réaliser sur le plan organisationnel et technique.
- Elles permettent d'identifier et d'évaluer les différentes alternatives permettant de réaliser ces objectifs.
- Elles permettent d'identifier les composants supportant ces buts. Etablir des liens entre les

objectifs organisationnels et les besoins du système est utile pour propager un changement des objectifs dans les besoins.
- Elles justifient et expliquent la présence de composants qui n'étaient pas forcément compréhensibles par les utilisateurs.
- Elles permettent d'assigner les responsabilités aux différents agents de l'organisation, et d'attribuer aux agents les rôles qui leur conviennent le mieux.
- Elles permettent de détecter et de résoudre les conflits pouvant survenir entre les agents dès les premières phases de conception du système.
- Elles permettent de vérifier la complétude de la spécification des besoins. Cette dernière est jugée complète si l'ensemble des buts peut être satisfait par l'ensemble des besoins de la spécification.

Axel van Lamsweerde [198] fournit un riche état de l'art portant sur l'utilisation de l'approche orientée but, et l'application des approches dirigées par les buts dans des projets réels.

1.3.3.1.2 Définition du modèle pivot

Nous proposons de spécifier les besoins des utilisateurs par un modèle orienté but, que nous avons défini comme un modèle *pivot* issus de trois approches répandues dirigées par les buts, qui sont : *KAOS*, *IStar* et *Tropos*.

1.3.3.1.2.1 Le modèle orienté but iStar (i*)

Le *framework i** est un Framework de modélisation standardisé proposé par *Eric Yu* [217] traitant la phase initiale d'analyse des besoins. Le framework i* distingue deux phases dans l'ingénierie des besoins :
- Phase initiale : qui vise à analyser et à modéliser les intérêts des utilisateurs et la manière de les intégrer dans différents systèmes et environnements.
- Phase finale : qui se concentre principalement sur la complétude, la consistance et la vérification automatique des besoins.

Le Framework i* fournit deux principaux modèles qui permettent de modéliser le système à développer et son environnement organisationnel.
- Le modèle de dépendance stratégique (Strategic Dependency (SD)) : décrit les relations entre les différents acteurs d'une organisation.
- Le modèle rationnel stratégique (Strategic Rationale (SR)) : fournit un niveau plus détaillé de modélisation, et décrit les buts et préoccupations des utilisateurs, et comment ils peuvent être implémentés par les différents systèmes et environnements. Nous nous intéressons particulièrement à ce deuxième modèle pour la définition de notre modèle pivot.

Le méta-modèle SR, présenté dans la figure 1.13 décrit les relations entre les éléments intentionnels utilisés dans le raisonnement de l'acteur. Le modèle est basé sur les principaux concepts de *but*, d'*acteur*, de *tâche*, de *ressource* et de *dépendance sociale*. L'acteur tient un rôle clé dans le Framework i*. Durant la phase d'analyse des besoins, le concepteur identifie les utilisateurs du système et les modélise en acteurs sociaux dépendants les uns des autres à travers les buts à réaliser, les tâches à effectuer et les ressources à utiliser.

Le modèle présenté en figure 1.14 présente un exemple (une instance) d'un modèle SR associé à l'acteur *Auteur*. Le but principal de l'auteur est que *son article soit publié*. Ce but est accompli

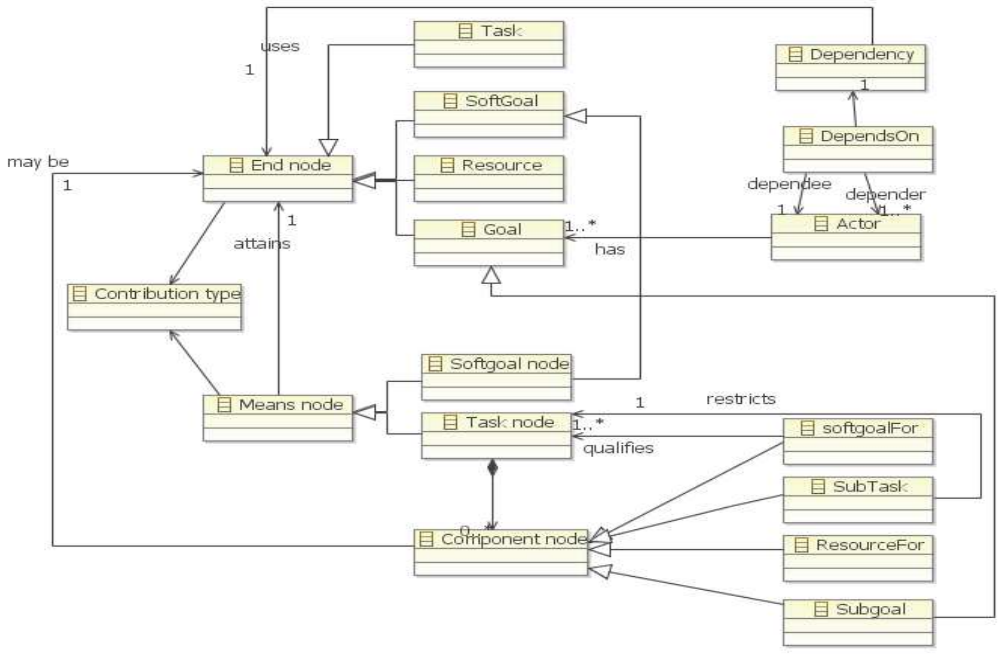

FIGURE 1.13 – Méta-modèle orienté but i* [211]

par la tâche *soumettre l'article*. Cette tâche est accomplie par deux sous-buts *article à soumettre* et *version finale à transmettre*. Les autres nœuds décrivent les buts et les tâches nécessaires pour accomplir ces deux sous buts.

1.3.3.1.2.2 Le modèle orienté buts Tropos

Tropos [190] est une méthodologie de développement de logiciels orientée agents. La méthodologie Tropos supporte trois phases du développement logiciel : l'analyse des besoins, la conception de l'architecturale du système (structure globale) et la conception détaillée du système. Tropos est basé sur les mêmes concepts du Framework i*, à savoir les concepts : *But*, *Acteur*, *Plan*, *Ressource* et *dépendance sociale*. Tropos fournit deux diagrammes :

- Diagramme d'acteurs : ce diagramme est l'équivalent du modèle SD du Framework i*. Le diagramme est fourni sous forme d'un graphe dont les nœuds représentent les acteurs du système et les arcs représentent les dépendances entre les acteurs.
- Diagramme de but : ce diagramme est l'équivalent du modèle SR du Framework i* et décrit les buts des utilisateurs. Le diagramme de but est représenté sous forme d'un graphe, dont les nœuds sont des buts, et les arcs sont les relations entre les buts. Nous nous sommes intéressés à ce diagramme pour la définition de notre modèle pivot.

La figure 1.15 présente un exemple du graphe de but proposé dans Tropos. L'exemple décompose le principal but *examiner article* en deux sous buts *assigner article aux lecteurs* et *collecter le rapport de lecture*. Le premier sous-but est également décomposé en deux sous buts : le but *envoyer article* qui est opérationnalisé par la tâche *envoyer article par courrier électronique* et le but *sélectionner les lecteurs* qui est décomposé en deux autres sous buts : *vérifier les compétences du lecteur* et *vérifier les conflits*. L'accomplissement de ces deux derniers sous buts

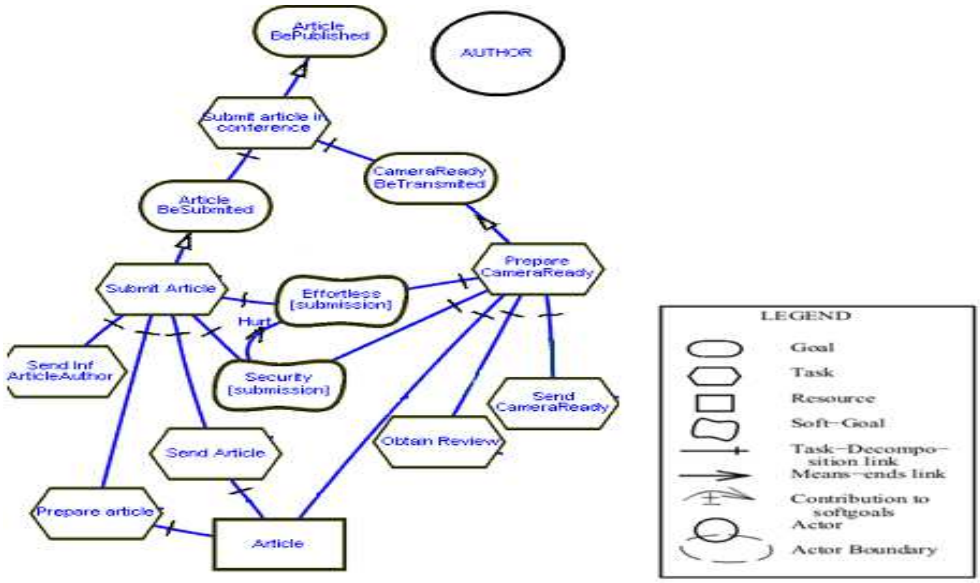

FIGURE 1.14 – Exemple d'un modèle orienté but i* [211]

contribue positivement à l'accomplissement du but *être équitable dans le processus d'affectation des articles.*

Ces notions sont formalisées par un modèle de buts Tropos [190], présenté dans la figure 1.16. Le concept central du modèle est le concept *but* représenté par la classe Goal. Les buts peuvent être analysés selon le point de vue de l'acteur, via les trois relations : *Mean-ends*, *Contribution*, et *Décomposition booléenne.* Les relations entre les buts sont : les relations ET/OU utilisées pour décomposer un but en sous buts et les relations qualitatives (+ et -) utilisées pour identifier les buts qui contribuent positivement ou négativement à l'accomplissement d'autres buts. Les buts non décomposables sont appelés des plans. Chaque diagramme de but est associé à un seul acteur.

1.3.3.1.2.3 Le modèle orienté buts KAOS

KAOS [52] est une méthodologie d'ingénierie des besoins dirigée par les buts, et signifie "Keep All Objects Satisfied". Le concept principal dans KAOS est le concept de But, dont la satisfaction nécessite la coopération d'agents pour configurer le système. KAOS fournit quatre modèles : un modèle de but, un modèle d'objet, un modèle de responsabilité et un modèle opérationnel. Nous nous sommes intéressés au modèle de but pour définir notre modèle pivot.

La figure 1.17 présente un exemple du graphe de but proposé dans KAOS. L'exemple décompose le principal but *qualité d'un article de publication* en plusieurs sous buts qui doivent tous être satisfaits pour satisfaire le but principal comme : *la définition du comité de programme*, la *réception des articles*, la *lecture des articles*, la *sélection des articles acceptés* et l'*édition finale.*

Le modèle de but de KAOS est illustré dans la figure 1.18. Le modèle comporte la principale classe But. Les buts sont liés entre eux par des relations de décomposition ET/OU. La décomposition des buts est arrêtée lorsqu'un sous but peut être exécuté par un agent. Les buts qui sont des feuilles dans le graphe de but sont appelés des *pré-requis.* Ils sont affectés à la responsabilité

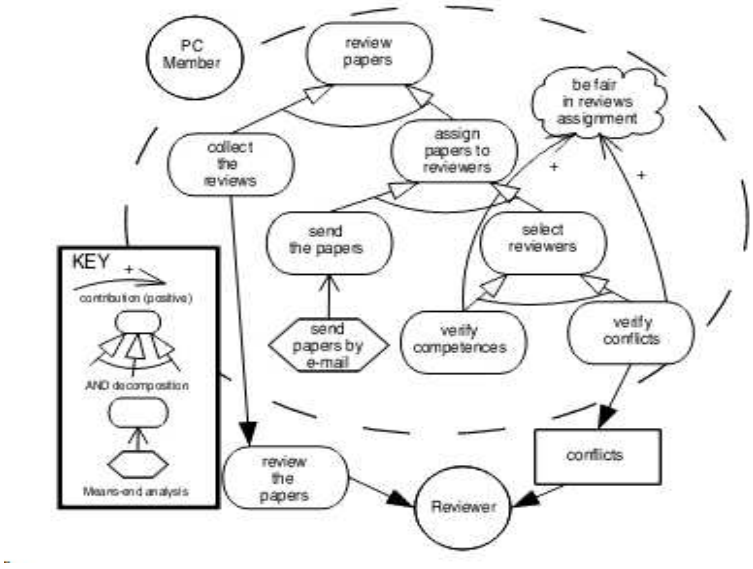

FIGURE 1.15 – Exemple d'un modèle orienté but Tropos [190]

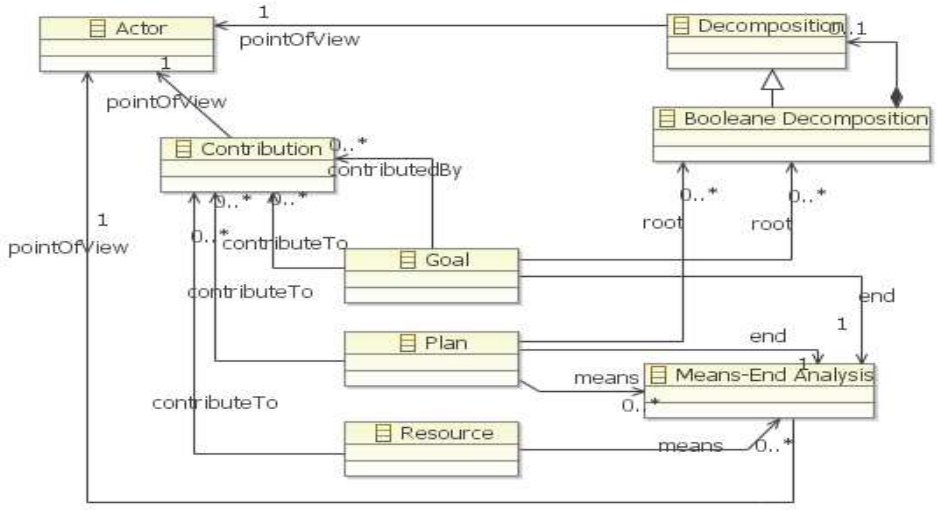

FIGURE 1.16 – Méta-modèle orienté but Tropos [190]

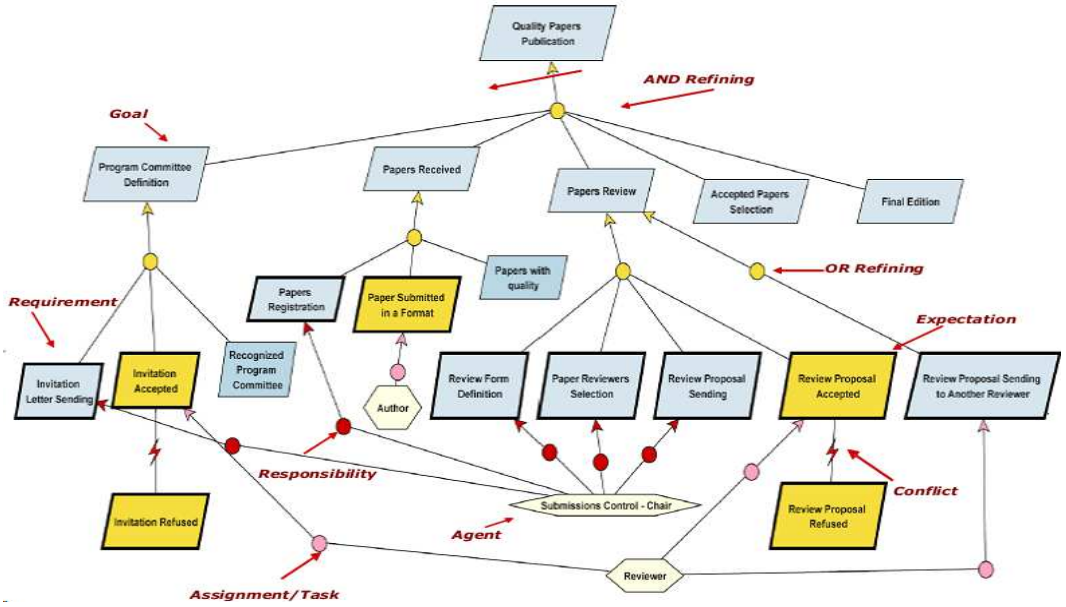

FIGURE 1.17 – Exemple d'un modèle orienté but KAOS [211]

d'un ou plusieurs agents. Lorsque ces buts sont assignés à un agent logiciel, on parle de besoin ou exigence. Les buts pré-requis sont rendus opérationnels par des actions. Les agents peuvent manipuler ou contrôler des objets identifiés lors de la définition des buts. Ces objets décrivent les entrées et sorties des actions. Les buts peuvent être fonctionnels (service) ou non fonctionnel (qualité de service).

1.3.3.1.2.4 Le modèle pivot

Après l'analyse des approches orientées but (*KAOS*, *IStar* et *Tropos*), nous avons proposé notre modèle de besoin dirigé par les buts, présenté en figure 1.19. Le noyau commun aux trois modèles précédents décrit les buts qui guident les diverses décisions de différents niveaux de l'entreprise. Les buts sont formulés à différents niveaux d'abstraction allant des buts stratégiques aux buts techniques (nommés aussi actions ou tâches ou plans). Pour représenter l'ensemble des buts de l'organisation, la représentation en graphe est utilisée. Dans cette représentation, les nœuds du graphe représentent les buts définis, et les arcs représentent des relations entre les buts. Afin de décomposer les buts stratégiques en sous buts jusqu'aux buts techniques, des relations ET/OU sont utilisées. Une *réduction ET* associe un but B à un ensemble de sous buts qui doivent tous être satisfaits pour que B le soit. Une *réduction OU* associe un but B à un ensemble de sous buts tels que la satisfaction de l'un d'entre eux assure la satisfaction de B. D'autres types de relations entre les buts sont définis comme les relations d'*influence* (ou de *contribution*). Dans ce type de relation, un but contribue partiellement (positivement ou négativement) à la satisfaction d'un autre but. Deux types de buts sont identifiés : les buts *fonctionnels* permettant de définir des fonctions du système et les buts *non fonctionnels* permettant de définir des qualités ou contraintes assurées par le système. Chaque but est associé à un acteur du système. Le modèle de but doit fournir le moyen de représenter les raisons pour lesquelles un acteur réalisent certaines tâches, poursuit un but, ou requiert une ressource. Nous avons, pour ce faire, lié chaque but à

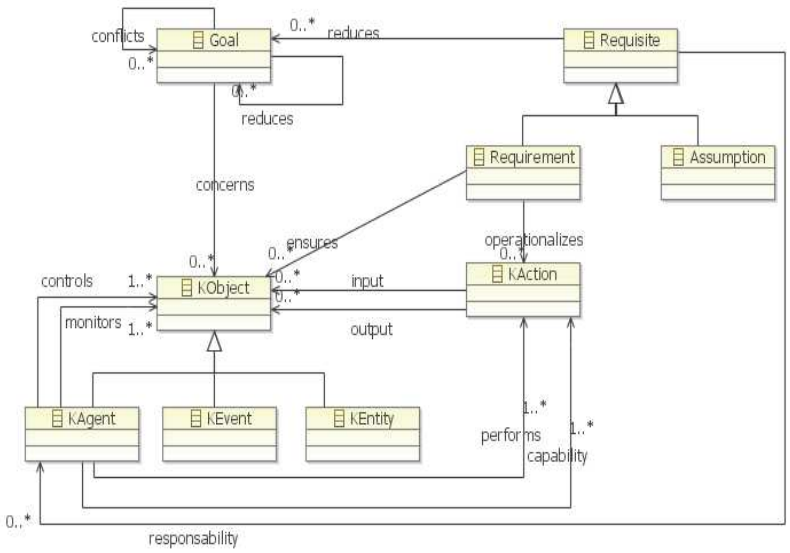

FIGURE 1.18 – Méta-modèle orienté but KAOS [83]

ses intensions définies par le résultat du but (ce qu'attend l'acteur à travers le but) et les critères du but (tous les objets ou ressources qui peuvent influencer ce but).

Afin d'illustrer nos propos, prenons l'exemple du but suivant extrait des besoins décisionnels de la spécification de SSB (après traduction en français) : *Ce besoin mesure l'augmentation des recettes des commandes selon différentes gammes de réductions pour un produit donné, selon les quantités commandées dans une année donnée. Notez que l'augmentation des recettes potentielles est égale à la somme de [Prix-étendu*Prix-remise].* Notre modèle se compose d'une classe principale Goal décrivant les buts. Un but est décrit par les attributs suivants : un identifiant id (*001*), un nom (*But G1.1*), une description, un objectif (*mesure*), un contexte (*Transactions commerciales*), une priorité et un booléen action. Le booléen *action* permet de distinguer un but décomposable d'un but représentant une action opérationnelle. Nous distinguons deux types de priorité : obligatoire et optionnel. L'attribut priorité peut aider à gérer les conflits entre les buts. Un but possède deux principales coordonnées liées aux but par une relation d'agrégation : un *Résultat* à analyser (*l'augmentation des recettes*) et un ou plusieurs *Critères* qui sont des concepts selon lesquels se fait l'analyse du but (*réductions*, *produit*, *Quantité* et *Année*). Nous avons enrichi le noyau commun identifié par les trois modèles de buts de KAOS, i* et Tropos, en spécialisant la classe Résultat. Le résultat du but peut être quantifié grâce à une *Métrique* semi-formelle ou formelle (*somme (Prix-étendu*remise)*). L'acteur représente toute entité interagissant avec le système pour répondre au but. L'acteur peut être une ou plusieurs personnes internes ou externes à l'organisation, un service de l'organisation ou un système logiciel. Les relations entre les buts sont de deux types : (i) les relations de *raffinement* ET/OU permettant de décomposer un but en plusieurs sous-buts. Par exemple, l'analyse de l'évolution des ventes est assurée par les trois premiers buts de la spécification SSB qui permettent d'analyser les recettes selon la date, les remises effectuées et les quantités vendues. (ii) Les relations de *contribution* représentent les relations d'influence positive et négative entre les buts. Par exemple, l'augmentation des ventes

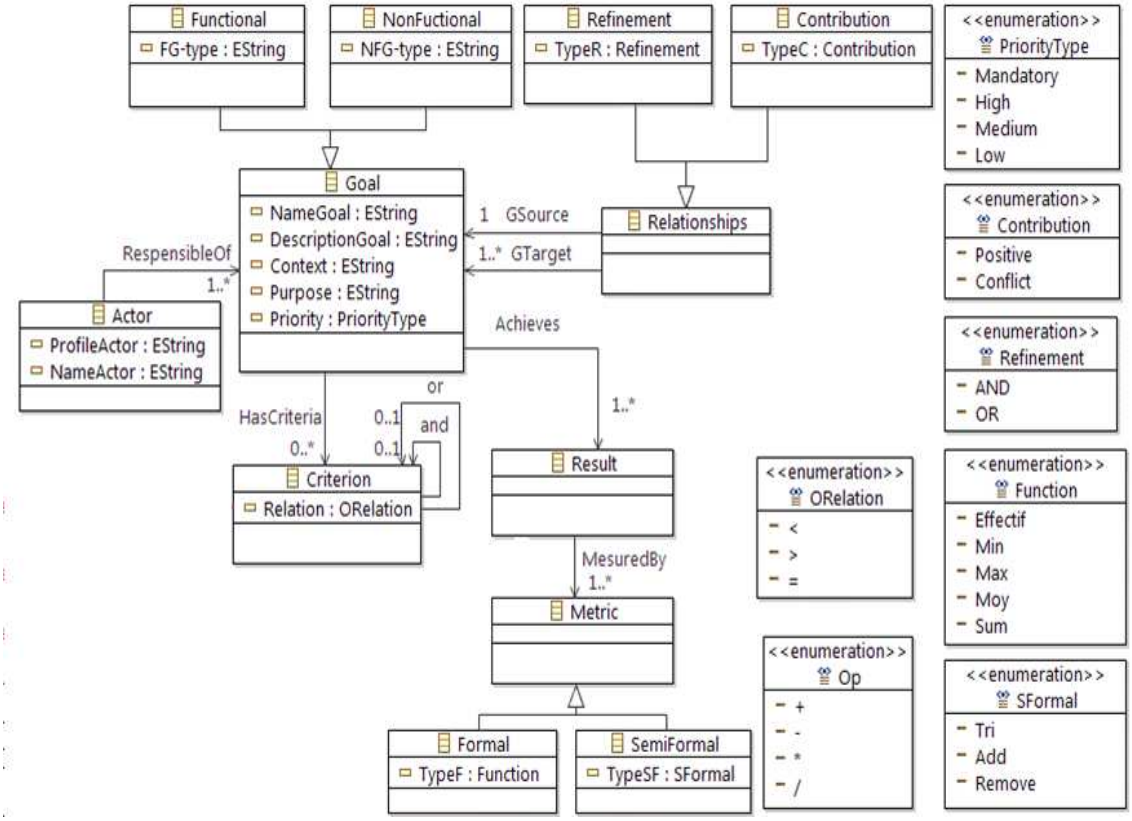

FIGURE 1.19 – Modèle de besoins proposé, dirigé par les buts.

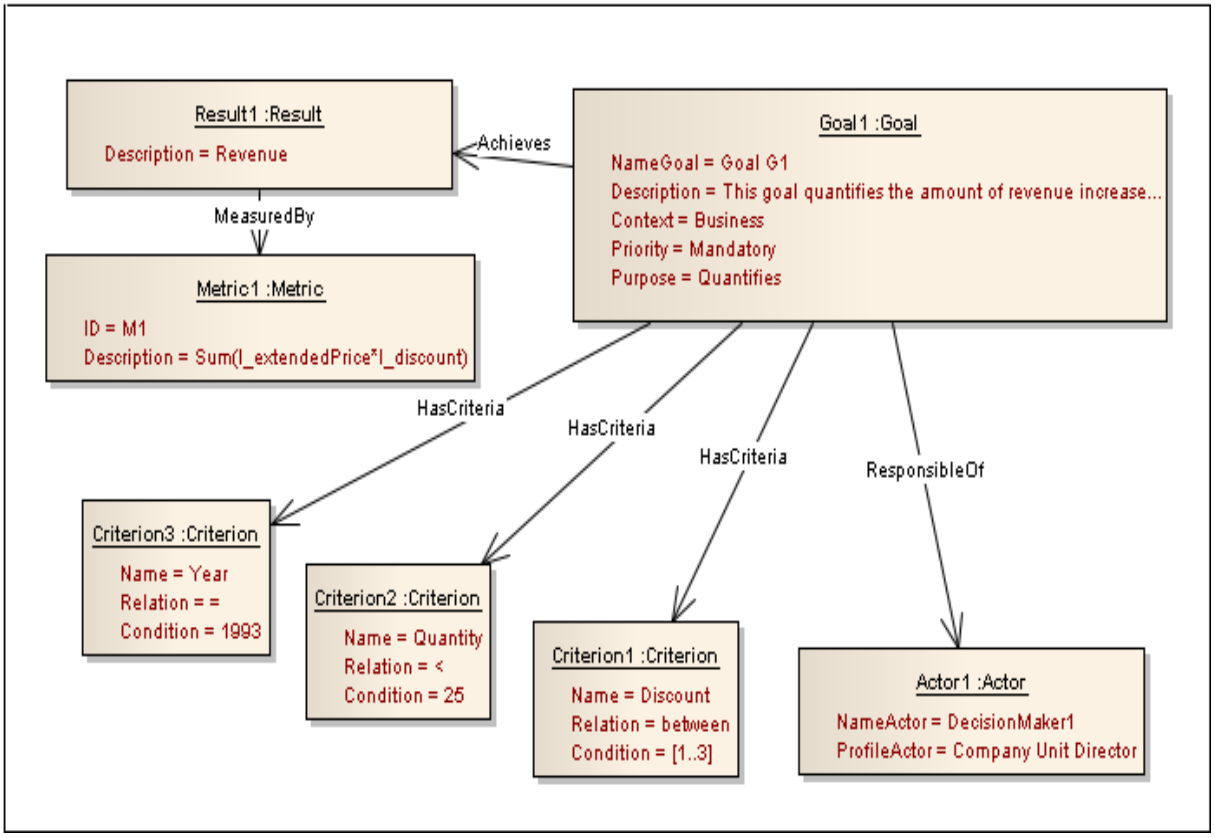

FIGURE 1.20 – Instanciation du modèle de but par le premier besoin de la spécification SSB

d'un produit peut influencer positivement sur le prix de ce produit. Les besoins définis peuvent être fonctionnels ou non-fonctionnels. Notons que plusieurs méthodes ont été proposées dans la littérature permettant l'acquisition des buts, l'identification des buts à partir des scénarios, et l'identification et découverte des relations de raffinement (ET/OU) et de contribution entre les buts [160]. Nous ne traitons pas ces aspects dans notre étude et nous considérons que le graphe des buts est validé.

Pour illustrer nos propos, nous avons instancié le modèle de but proposé en utilisant le premier besoin de la spécification du banc d'essai SSB comme présenté dans la figure 1.20.

La figure 1.21 illustre l'ensemble des buts de la spécification SSB présentés sous forme d'un graphe de buts. L'ensemble des besoins et requêtes du banc d'essai SSB sont décrits dans l'annexe IV.

1.3.3.2 Le modèle de l'ontologie

Une première formalisation proposée par *Guy Pierra* [151] au laboratoire LIAS représente une ontologie par le quadruplet $\mathcal{O} :< \mathcal{C}, \mathcal{P}, \mathcal{S}ub, \mathcal{A}pplic >$ où :

- $\mathcal{C}$: l'ensemble des classes utilisées pour décrire les concepts d'un domaine donné. Chaque classe est associée à un identifiant universel globalement unique (GUI).
- $\mathcal{P}$: l'ensemble des propriétés utilisées pour décrire les instances de l'ensemble des classes $\mathcal{C}$. $\mathcal{P}$ définit toutes les propriétés susceptibles d'être présentes dans une BD. Chaque propriété

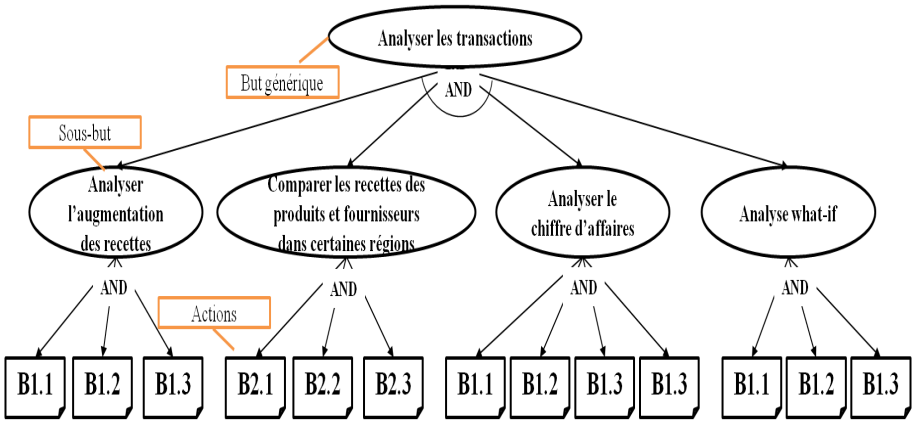

FIGURE 1.21 – Graphe des buts de la spécification SSB

est associée à un identifiant universel globalement unique.

- $\mathcal{S}ub : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ est la relation de subsomption [7] (*is-a*) qui associe à chaque classe c de l'ontologie ses classes subsumées directes. $\mathcal{S}ub$ définit un ordre partiel sur $\mathcal{C}$.
- $\mathcal{A}pplic : \mathcal{C} \rightarrow 2^{\mathcal{P}}$ associe à chaque classe de l'ontologie les propriétés qui sont applicables pour chaque instance de cette classe. Les propriétés qui sont applicables sont héritées via la relation *is-a*.

Cette formalisation a été proposée dans le cadre du projet *PLIB* qui formalise les ontologies canoniques représentées par le langage *PLIB*. Afin de généraliser cette formalisation pour les ontologies non canoniques, nous avons dû l'étendre par les définitions non canoniques. Pour ce faire, nous avons introduit un nouvel opérateur *Ref* qui représente les références entre les concepts et les propriétés de l'ontologie.

Le langage *OWL* présente actuellement le standard recommandé par le W3C pour la définition des ontologies canoniques et non canoniques. Le langage OWL est basé sur le formalisme des *logiques de descriptions* (LD), qui présente une famille des logiques du premier ordre. Les expressions définies pour les ontologies OWL basées sur les logiques de description fournissent un ensemble d'opérateurs couvrant plusieurs formalismes ontologiques tels que *RDF*, *RDFS* et *DAML+OIL*. Par conséquent, nous définissons le modèle de l'ontologie en nous basant sur le formalismes des LD. Nous commençons par présenter ce formalisme en décrivant sa syntaxe et sa sémantique. Nous proposons ensuite notre formalisation de l'ontologie.

1.3.3.2.1 Formalisation du modèle de l'ontologie

Le formalisme des LD définit des logiques destinées à représenter des connaissances structurées et à raisonner sur ces connaissances [12]. Dans les LD, les connaissances du domaine sont décrites à travers des *concepts* (Exp. les concepts *Customer*, *Supplier*, *Order*, dans le schéma de la figure 1.10) et des *rôles* (Exp. *supply* et *timeOrder*). Les concepts décrivent un ensemble d'individus (instances), et les rôles décrivent des relations binaires entre les individus (ou entre un individu et un domaine de valeurs comme les entiers, réels, etc.). Deux types de concepts sont distingués : les concepts *atomiques* (concepts primitifs) et les concepts de *descriptions* (concepts

7. $2^{\mathcal{C}}$ représente l'ensemble des parties de $\mathcal{C}$

définis), qui sont définis par d'autres concepts en utilisant des *constructeurs* des LD.

Il existe plusieurs familles de LD. Chaque famille utilise un sous-ensemble de *constructeurs*. Les constructeurs qui définissent le fragment basique AL (Attributive language) sont [12] : C,D → A (concept atomique)| ⊤ (concept universel) | ⊥ | ¬A (négation atomique)| $C \sqcap D$ (intersection) | ∀ R.C (restriction de valeur) | $\exists R.\top$ pour la quantification existentielle limitée (A : concept atomique, R et S : rôles atomiques, C et D : concepts de descriptions).

Une base de connaissances en LD est composée d'une TBOX (Terminological Box) décrivant les connaissances *intensionnelles* du domaine sous forme d'axiomes, et d'une ABOX (Assertion Box) décrivant les connaissances *extensionnelles* du domaine (les instances). Par exemple, Order(*Order#1*) indique que *Order#1* est une instance du concept *Order*. Les axiomes terminologiques peuvent être définis en utilisant des inclusions : C ⊏ D (R ⊏ S) ou des équivalences : C ≡ D (R ≡ S) (Exp. Order ⊏ Transaction décrit le concept *Order* comme sous concept de *Transaction*).

Des langages plus expressifs sont obtenus en ajoutant d'autres constructeurs du fragment AL. Par exemple, le langage ALC (C pour complément) est obtenu en ajoutant la négation de concepts (¬ C). Le langage ALU est obtenu en ajoutant l'union des concepts (C ⊔ D). Le langage ALE est obtenu en ajoutant la quantification universelle (∀ R.C). L'extension du langage AL par un sous-ensemble des constructeurs cités ci-dessus donne un langage nommé comme suit : AL[U][E][N][C], où chaque lettre ajoutée dénote la présence du constructeur correspondant [12]. Le langage d'ontologie OWL est basé sur la famille ALC des LD. OWL-DL est basé sur le fragment SHOIND(D) et OWL-Lite est basé sur le fragment SHIF(D). Par exemple, le fragment SHIF correspond à la logique de description ALC enrichie par les constructeurs de restrictions de fonctionnalités, de hiérarchie de rôles et des rôles transitifs et inverses. Le fragment SHOIND(D) fournit la possibilité d'utiliser les instances et des restrictions de valeurs. Nous décrivons dans ce qui suit les principaux constructeurs de LD et leurs équivalents en langage OWL [12]. Le tableau 1.1 résume ces correspondances.

1. Intersection (∩, `owl:intersectionOf`) permet de déclarer une nouvelle classe comme étant l'intersection de deux ou plusieurs classes. Par exemple, la classe *Man* peut être définie comme l'intersection des classes *Human* et *Male*. ($Person \equiv Human \cap Male$).
2. Union (∪, `owl:UnionOf`) permet de déclarer une nouvelle classe comme étant l'union de deux ou plusieurs classes. Par exemple, on pourrait déclarer que la classe *Person* est l'union des classes *Man* et *Woman*. ($Person \equiv Man \cup Woman$).
3. Complément (¬, `owl:complementOf`) permet de déclarer une classe comme étant le complémentaire de deux classes. Par exemple, on pourrait déclarer que la classe *Woman* est le complémentaire de la classe *Man* dans la classe *Person* ($Woman \equiv Person - Man$).
4. Énumération (`owl:oneOf`) permet de déclarer une classe par extension. La population d'instances est listée dans un ensemble.
5. Restriction (`owl:Restriction`) permet de déclarer une nouvelle classe à partir d'une autre en définissant des conditions sur les instances de la classe. Ces conditions peuvent être des restrictions sur le domaine de valeurs d'une ou plusieurs propriétés de la classe ou sur la cardinalité des propriétés.
   - Quantification Universelle (∀ $P.C_1$, `owl:allValuesFrom`) permet de déclarer une nouvelle classe en imposant pour chaque instance de la classe ayant des instances via la propriété

Constructeurs	Logique de description	OWL
Intersection	$C_1 \cap ... \cap C_n$	`intersectionOf`
Union	$C_1 \cup ... \cup C_n$	`unionOf`
Complément	$\neg C_1$	`complementOf`
Énumération	$\{o_1, ..., o_n\}$	`oneOf`
Quantification Universelle	$\forall\ P.C_1$	`allValuesFrom`
Quantification existentielle	$\exists\ P.C_1$	`someValuesFrom`
Egalité de valeur	$\exists\ P.\{o_1\}$	`hasValue`
Cardinalité minimale	$\leq nP.C_1$	`minCardinality`
Cardinalité maximale	$\geq nP.C_1$	`maxCardinality`
Cardinalité exacte	$= nP.C_1$	`cardinality`

TABLE 1.1 – Correspondances des constructeurs du langage OWL avec ceux de la logique de description

indiquée, que les valeurs de la propriété soient toutes membres de la classe désignée. Par exemple, on pourrait déclarer que la classe *Father* est la classe *Person* avec restriction de la propriété *father* à la classe *Man* ($Father \equiv \forall father.Man$).

- Quantification existentielle ($\exists\ P.C_1$, `owl:someValuesFrom`) est similaire à la précédente, à la différence qu'elle déclare une nouvelle classe en imposant pour chaque instance de la classe ayant des instances via la propriété indiquée, qu'au moins une des valeurs de la propriété soit membre de la classe désignée.
- Egalité de valeur ($\exists\ P.\{o_1\}$, `owl:hasValue`) permet de déclarer une nouvelle classe en imposant la valeur spécifiée à toutes les instances de la classe en question. Par exemple, on pourrait déclarer que la classe *Woman* est la classe *Person* avec la propriété *sex* égale à *Female* ($Woman \equiv \exists sex.Female$).
- Cardinalité minimale ($\leq nP.C_1$, `owl:minCardinality`) permet de déclarer une nouvelle classe en fixant une borne inférieure au nombre d'instances dans une propriété.
- Cardinalité maximale ($\geq nP.C_1$, `owl:maxCardinality`) permet de déclarer une nouvelle classe en fixant une borne supérieure au nombre d'instances dans une propriété.
- Cardinalité exacte ($= nP.C_1$, `owl:cardinality`) permet de déclarer une nouvelle classe en indiquant exactement le nombre d'instances dans une propriété.

D'un point de vue *sémantique*, les concepts en LD sont interprétés comme des sous ensemble d'un domaine, et les rôles sont interprétés comme des relations binaires sur le domaine. Une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ définie sur l'ensemble $\mathcal{A}$ de concepts atomiques et l'ensemble $\mathcal{P}$ des rôles atomiques, consiste en un ensemble $\Delta^{\mathcal{I}}$ fini et non vide (le domaine de $\mathcal{I}$) et une fonction $.^{\mathcal{I}}$ (la fonction d'interprétation de $\mathcal{I}$) qui fait correspondre chaque concept atomique $A \in \mathcal{A}$ à un sous ensemble $A^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$ (l'ensemble des instances de A), et chaque rôle atomique $P \in \mathcal{P}$ à un sous ensemble $P^{\mathcal{I}}$ de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ (l'ensemble des instances de P). La fonction d'interprétation est étendue pour les concepts définis. Par exemple, $(C_1 \sqcap C_2) = (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}}$

Nous fournissons à présent la formalisation du modèle de l'ontologie (MO), basée sur le formalisme des LD, définie comme suit : MO :<C, R, Applic(C), Sub(C), Ref (C), Ref'(R), Formalisme>, où :

- C représente les classes du modèle ontologique.
- R représente les propriétés (rôles) du modèle ontologique.

- $Applic : C \rightarrow 2^R$ est une fonction qui permet de lier chaque classe aux propriétés qui lui sont attachées.
- $Sub : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ est la relation de subsomption qui, à chaque classe $c \in C$ de l'ontologie, associe ses classes subsumées directes.
- $Ref(C) : C \rightarrow (operateur, Exp(C, R))$ est une fonction définissant les classes de la TBOX. *Ref* associe à chaque classe un *opérateur* (d'inclusion ou d'équivalence) et une *expression Exp* sur d'autres classes et propriétés. Ces expressions utilisent l'ensemble des constructeurs de la LD cités. *Exp* peut être la fonction d'identité qui associe à une classe la même classe (la classe se définit par elle-même comme la plus grande classe de la hiérarchie Thing). Par exemple, la classe Order de l'ontologie SSB est définie comme sous classe de la classe Thing, comme suit : Ref(Order) = ($\sqsubset$, Thing).
- $Ref'(R) : R \rightarrow (operateur, Exp(C, R))$ est une fonction définissant les propriétés de la TBOX. *Ref'* associe à chaque propriété un opérateur (d'inclusion ou d'équivalence) et une expression sur d'autres classes et propriétés.
- *Formalisme* est comme son nom l'indique le formalisme du modèle ontologique adopté.

Cette formalisation est générique pour les langages d'ontologies usuels et pour les modèles conceptuels conventionnels (E/A ou UML). *Calvanese et al.* [40] ont proposé des algorithmes permettant de traduire un modèle de diagramme de classes UML ou un modèle E/A en assertions de LD. Par exemple, le modèle E/A instancie la formalisation comme suit :
E/A :<Entités, Attributs et associations, Ref : <Opérateur : assertions d'inclusion, Exp : expression utilisant les constructeurs <inverse de role, role functionnel and restriction de valeur>, E/A>.

Pour les besoins de notre méthode de conception, nous utiliserons le langage OWL-DL, qui est basé sur le fragment SHOIN(D) des LD. Une ontologie OWL-DL instancie la formalisation comme suit OWL-DL : <*Classes*, *Propriétés Object et propriétés DataType*, *Applic*, *Sub :subClassOf*, *Ref* et *Ref'* définies en se limitant aux constructeurs de la logique de description *SHOIN(D)*, *OWL-DL*>.

Pour reprendre notre cas pratique, la première tâche a été de représenter le modèle conceptuel SSB (représenté par un diagramme de classes UML) par une ontologie OWL en suivant les étapes suivantes définies par *Calvanese et al.* [40] :

- Chaque classe du digramme est représentée par un concept atomique.
- Chaque attribut att de type T de la classe C est représenté par un rôle atomique comme suit :
  - C $\sqsubseteq \forall$ att.T (représente le type T de att pour C)
  - C $\sqsubseteq$ ($\geq$ i att) $\sqcap$ ($\leq$ j att) (représente la multiplicité [i..j] de att). Lorsque j est *, la deuxième parenthèse est omise.
  - Lorsque la multiplicité est de [0..*] toute l'assertion est omise. Lorsque la multiplicité est de [1..1] l'assertion devient de la forme : C $\sqsubseteq \exists$att $\sqcap$ ($\leq$ 1 att)
- Chaque association A binaire entre deux classes est représentée par un rôle.
- Chaque classe d'association non binaire est représentée par un concept connecté à tous ses composants par un rôle.
- Les relations d'hériage (C1, C2, Ck hérite de C) sont traduites en utilisant l'opérateur d'inclusion : Ci $\sqsubseteq$ C.

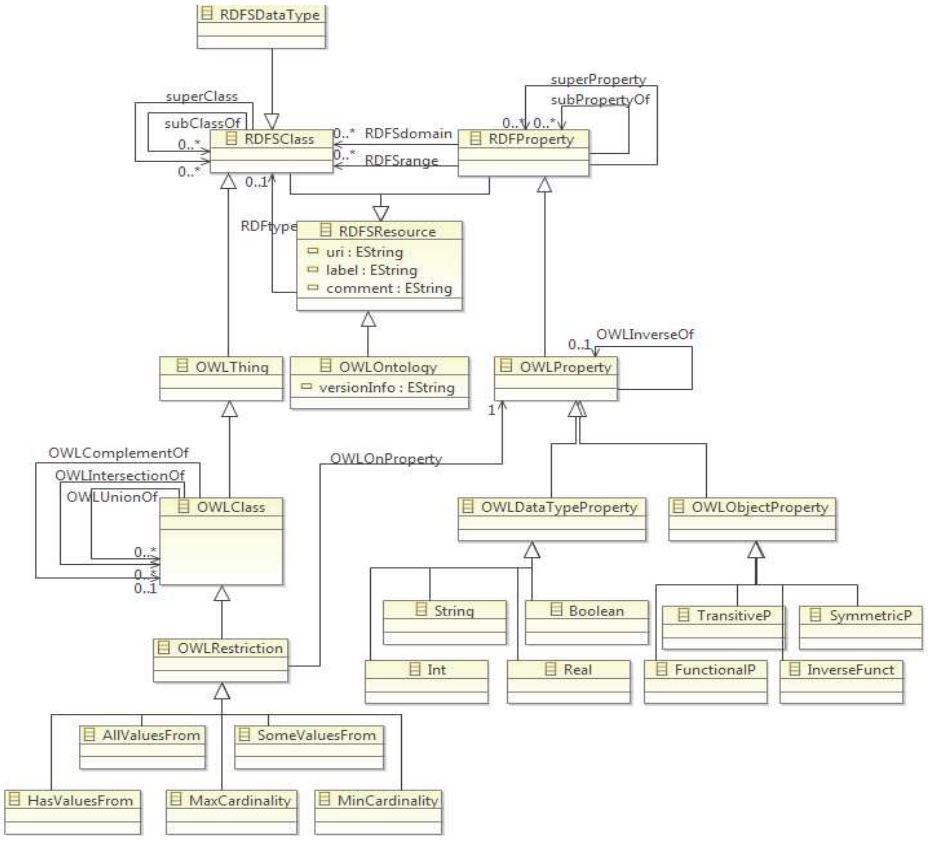

FIGURE 1.22 – Méta-modèle OWL

Si la relation de généralisation est disjointe : Ci $\sqsubseteq \neg$Cj.

Si la relation de génération est complète : C $\sqsubseteq$ C1 $\sqcup$ C2 $\sqcup$ ... $\sqcup$ Ck.

Chaque partie du digramme de classes est ainsi traduite par un ensemble d'assertions d'inclusion.

1.3.3.2.2 Représentation du modèle de l'ontologie

D'un point de vue modèle de représentation, le standard Meta-Object Facility (MOF2) de l'Object Management Group (OMG) fournit un métamodèle OWL décrivant les principaux constructeurs d'une ontologie OWL. Ce modèle est présenté dans la figure 1.22. Le métamodèle OWL comporte les principales classes suivantes : *OWLClass*, *DataTypeProperty* et *ObjectProperty*, représentant respectivement les classes, les propriétés de type attribut et les propriétés de type relations. La subsomption des classes est représentée par la relation réflexive *subclassOf*.

Les propriétés possèdent éventuellement un domaine (classe source) et un rang (classe cible ou type de donnée). Une propriété peut être monovaluée (*FunctionalProperty*) i.e elle admet au plus une valeur pour toute instance de son domaine. Elle peut être multivaluée i.e elle a une collection de valeurs (éventuellement vide) pour une instance donnée de son domaine. Elle peut être symétrique (*SymmetricProperty*) ou transitive (*TransitiveProperty*).

Différentes classes sont représentées pour définir les constructeurs de LD (*ComplementClass*, *EnumeratedClass*, *IntersectionClass*, *OWLRestriction*, *UnionClass*). La classe *Individual* détermine les instances des classes.

1.3.4 Méthode de conception proposée

Cette section présente la méthode de *conception du schéma de l'ED* selon les étapes du cycle de conception qui sont : la définition des besoins, la modélisation conceptuelle, la modélisation logique et la modélisation physique. La phase ETL sera traitée au prochain chapitre pour alimenter le schéma de l'ED défini, par les données des sources.

1.3.4.1 La définition des besoins

La première contribution présentée dans ce chapitre consiste à spécifier les besoins des utilisateurs au niveau ontologique. Dans les systèmes d'informations de manière générale, les besoins des utilisateurs permettent de constituer le cahier des charges de l'application à développer. En poussant cette réflexion, l'ontologie peut être vue comme le cahier des charges couvrant l'ensemble des besoins du domaine, qui est défini de manière formelle et consensuelle. Nous montrons dans cette section que la spécification des besoins au niveau ontologique permet de valider le modèle des besoins, et d'effectuer du raisonnement sur les besoins.

Les buts des utilisateurs sont spécifiés au niveau ontologique où nous avons défini une connexion entre les coordonnées de chaque but (résultat et critères) et les classes et propriétés de l'ontologie de domaine. Nous rappelons que l'ontologie représente un modèle partagé par l'ensemble des sources. L'ontologie permet au concepteur d'avoir une vue globale du domaine à partir de laquelle il peut aisément spécifier son cahier de charge en sélectionnant les concepts et propriétés qui représentent les besoins. Les besoins sont exploités dans ce processus afin de déterminer les informations les plus pertinentes à stocker dans le modèle de l'ED. La figure 1.23.(A) représente un fragment (le noyau commun aux différents formalismes d'ontologies) du modèle d'ontologie auquel est connecté le modèle de besoins (figure 1.23.(B)).

La figure 1.24 présente un exemple de l'instanciation du modèle de but connecté au modèle de l'ontologie. Le modèle de but est instancié par le premier besoin de la spécification SSB. Le modèle de l'ontologie est instancié par l'ontologie définie à partir de la spécification SSB.

En prenant le cas d'une ontologie OWL, nous avons défini le modèle de but proposé au niveau ontologique en étendant le méta-modèle OWL. Concrètement, ceci se fait par la création de nouvelles métaclasses (comme *But*, *Résultat*, *Critère*, etc), et en liant les coordonnées des buts (*Résultat* et *Critère*) avec la métaclasse *rdfs :Class* de l'ontologie OWL (figure 1.25).

1.3.4.2 La modélisation conceptuelle

La phase de modélisation conceptuelle comporte trois étapes :
- l'extraction d'une ontologie locale à l'ED reflétant les besoins des utilisateurs,
- le raisonnement ontologique sur le modèle de besoins,
- la définition du modèle conceptuel multidimensionnel de l'ED.

1.3.4.2.1 Définition de l'ontologie de l'ED

L'ontologie locale à l'ED (OED) est définie par extraction des classes et propriétés ontologiques de l'ontologie partagée (OP), intervenant dans la définition des besoins. Cette ontologie locale OED représente le schéma conceptuel de l'ED à développer et décrit la sémantique de ses entités.

Trois scénarios d'extraction sont possibles :

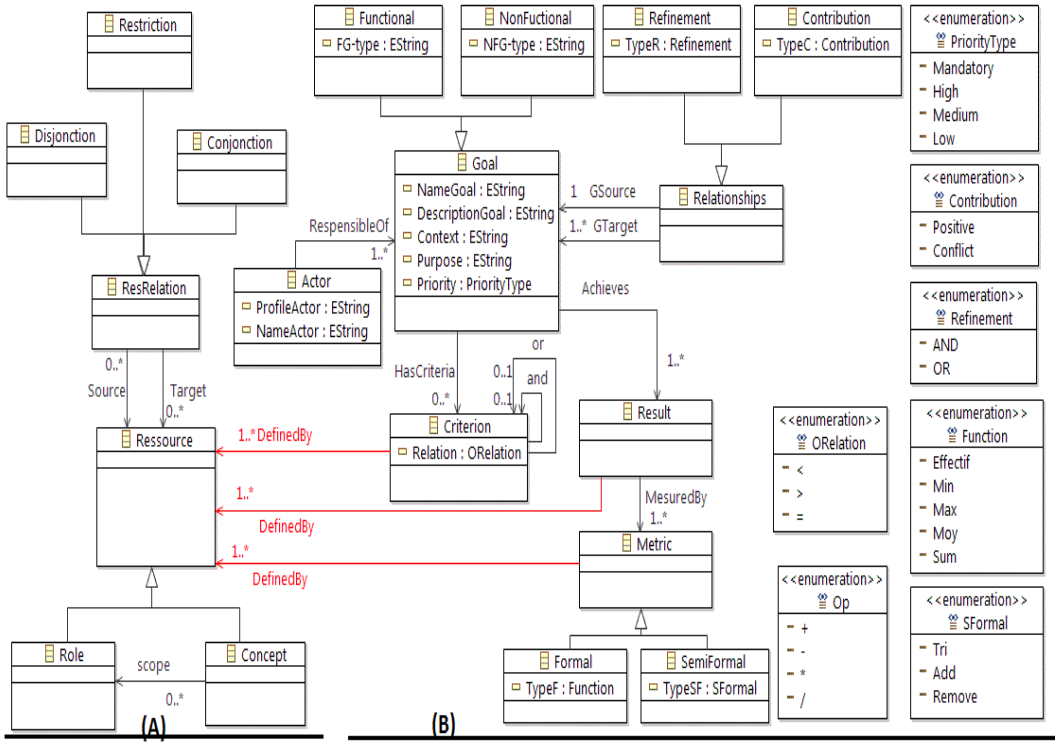

FIGURE 1.23 – Le modèle des besoins connecté au métamodèle de l'ontologie.

- $OED = OP$: l'OP correspond exactement aux besoins des utilisateurs.
- $OED \subset OP$: l'OED est extraite à partir de l'ontologie partagée et couvre tous les besoins des utilisateurs.
- $OED \supset OG$: L'ontologie partagée ne couvre pas tous les besoins spécifiques des utilisateurs. L'ontologie locale est extraite, et peut être étendue par de nouvelles classes ou propriétés.

1.3.4.2.2 Raisonnement sur le modèle de besoins

Les mécanismes de raisonnement des ontologies sont basés sur les logiques de description et sont de différents types. Nous utilisons trois principaux mécanismes de raisonnement :

1. Modularité ontologique
2. Validation du modèle
3. Propagation des relations de contributions

Modularité ontologique : le premier mécanisme se rapporte à la *modularité ontologique*. Dans le deuxième et troisième scénario d'extraction cité ci-dessus, nous définissons l'ontologie locale OED à partir de l'ontologie partagée par extraction des classes et propriétés ontologiques intervenant dans la définition des besoins. Cette extraction se fait selon une méthode de modularité, qui permet d'extraire une ontologie à partir d'une autre, tout en assurant la *complétude* et la *cohérence* de l'ontologie extraite. Prenons l'exemple d'une ontologie OWL dont les expressions des concepts et propriétés sont basées sur la logique de description. Une ontologie OWL extraite

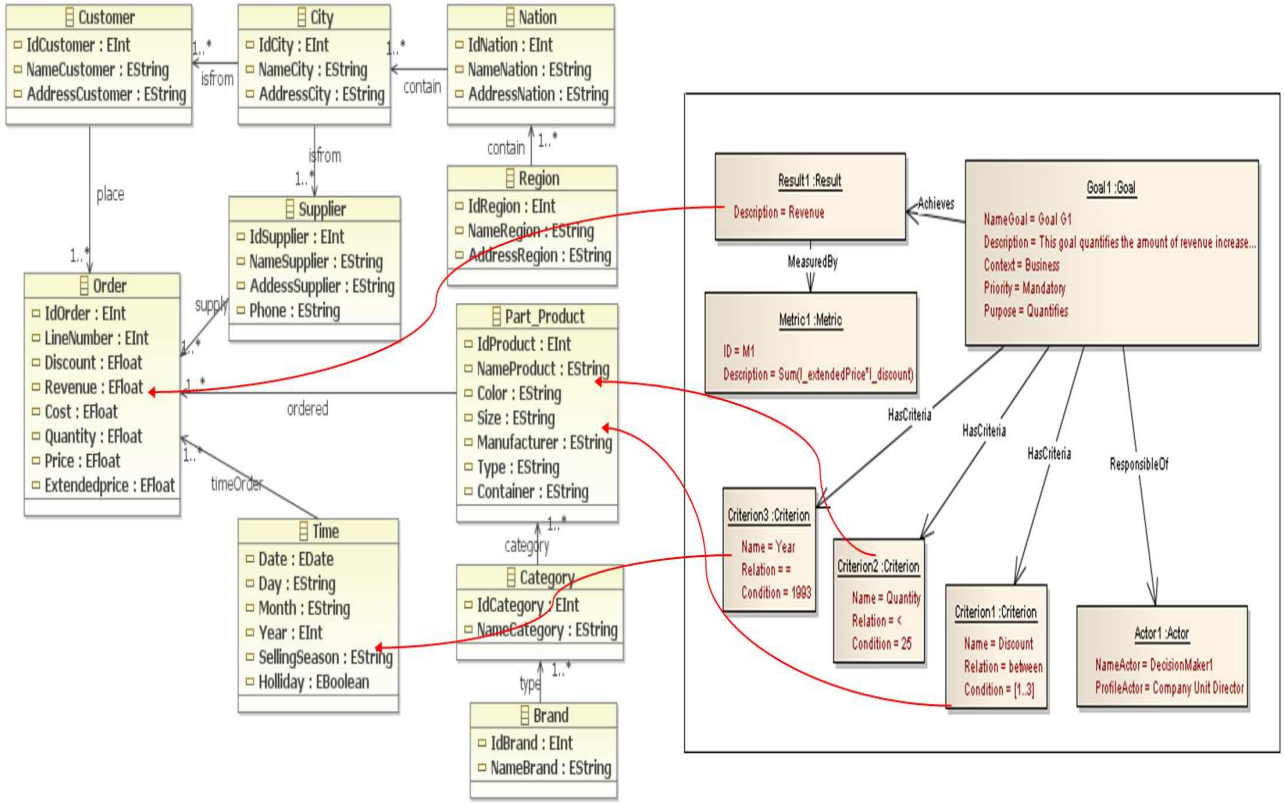

FIGURE 1.24 – Instanciation des modèles de l'ontologie et du modèle de but

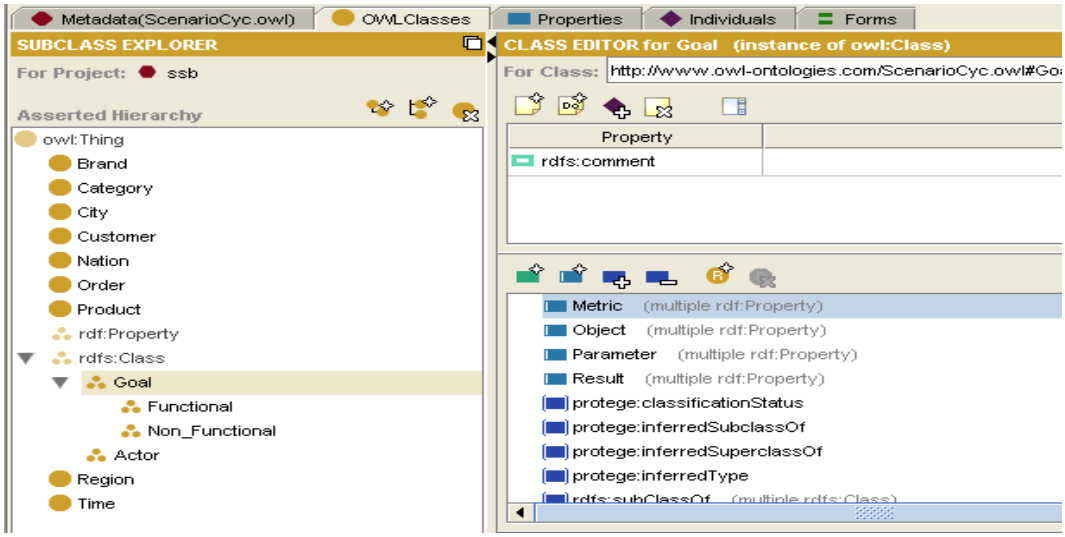

FIGURE 1.25 – Modèle de but étendant le modèle d'ontologie OWL (sous Protégé)

comme module à partir d'une autre ontologie OWL ne doit contenir aucun fait logique contredisant un autre fait (*cohérence* de l'ontologie). De plus, les faits que décrit cette ontologie module doivent être complètement définis au sein de cette ontologie (*complétude* de l'ontologie). Plusieurs méthodes de modularité ont été proposées dans la littérature, pour les différents langages ontologiques. Dans une ontologie PLIB, la modularité est assurée par l'opérateur *Case-of*. D'autres opérateurs de modularité ont été proposés pour les ontologies OWL. Nous avons opté pour une méthode de modularité basée sur le formalisme des LD et pour laquelle un plugin de modularité a été implémenté sous l'éditeur d'ontologie Protégé [75]. La formalisation de la modularité est basée sur les deux définitions suivantes :

Définition 1 *Soit O l'ontologie importée. Soit* O_1*' ⊆ O'.*
O_1*' est un module de O' importée dans l'ontologie O si O ∪ O' est une extension conservatrice (S-conservative extension) de* O_1*' ∪ O'.*

Définition 2 *O est une extension conservatrice de* O_1 *si les axiomes représentés dans O ne rajoutent pas de nouvelles implications (axiomes) autre que la signature de* O_1*.*
Soit $O_1 \subseteq O'$*, soit S est la signature (ensemble de termes à extraire dans le module). Soit Sig(.) la fonction qui lie les axiomes à la signature.*
O est une S-Conservative Extension de O_1 *si :*
$\forall \alpha$ *tel que Sig(*α*)* $\subseteq S$*,* $O \models \alpha$ *si et seulement si* $O_1 \models \alpha$*.*

En étendant la première formalisation de l'ontologie partagée, l'ontologie de l'ED (OED) sera définie formellement de la manière suivante : $O :< C_{ED}, R_{ED}, Applic_{ED}, Sub_{ED}, Ref_{ED}, Ref'_{ED}$, Formalism, Besoin> tel que *Besoin* est l'ensemble des besoins des utilisateurs définis en utilisant les classes et propriétés ontologiques : $Besoin \in 2^{C_{ED}} U 2^{R_{ED}}$.

Validation du modèle : d'autres mécanismes de raisonnement sont utilisés dans notre méthode. Selon la terminologie utilisée en LD, une ontologie OWL est constituée des deux composants : la TBOX $\mathcal{T}$ décrivant les concepts et propriétés et la ABOX $\mathcal{A}$ décrivant les instances. Les mécanismes de raisonnements utilisés sont les suivants :

- Vérification des relations de subsomption : un concept C est subsumé par un concept D (assertion d'inclusion) par rapport à une TBox $\mathcal{T}$ si l'extension de C est incluse dans l'extension de D dans chaque modèle $\mathcal{I}$ de $\mathcal{T}$. De manière formelle, une interprétation $\mathcal{I}$ satisfait une assertion d'inclusion ($C \sqsubseteq D$) si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
  Une interprétation $\mathcal{I}$ est un modèle pour la TBOX $\mathcal{T}$ si elle satisfait toutes ses assertions d'inclusion.
  Cette inférence possède plusieurs avantages comme la détection des erreurs de modélisation dans une ontologie, la classification automatique de la hiérarchie de classes et l'inférence d'une nouvelle hiérarchie. Ce mécanisme de raisonnement peut être utilisé pour identifier les classes équivalentes et les besoins équivalents (représentés eux même par des classes de l'ontologie). Deux classes C et D sont équivalentes si chaque classe subsume l'autre ($C \sqsubset D$) et ($D \sqsubset C$). Identifier les classes équivalentes permet d'éliminer la redondance et donc de réduire la complexité du modèle.
- Vérification de la satisfiabilité d'un concept : ceci implique de déterminer si un modèle de $\mathcal{T}$ existe dans lequel le concept possède une extension non vide. En d'autres termes, un

concept non satisfiable ou inconsistant est un concept qui ne peut avoir d'instances. La présence de concepts non satisfiables est généralement dûe à des erreurs de modélisation, cette inférence permet donc de détecter et corriger ces erreurs. La formalisation de ce type de raisonnement se définit comme suit :
un concept est satisfiable dans la TBOX $\mathcal{T}$ s'il existe un modèle $\mathcal{I}$ de $\mathcal{T}$ tel que : $C^{\mathcal{I}} \neq \emptyset$.

– Consistance de la base de connaissances : consiste à s'assurer que l'ontologie ne contient aucun fait contradictoire. Ceci est vérifie lorsque la TBox et la ABox ont un modèle commun non vide.

Tous ces raisonnements sont supportés par la plupart des raisonneurs actuels comme *Pellet*, *Racer* ou *Fact*.

Propagation des relations de contributions : Le troisième mécanisme de raisonnement utilisé consiste à définir des règles logiques pour raisonner sur les buts spécifiés sur l'ontologie. Plus précisément, nous propageons les relations d'influence (contribution) entre les buts. Ces relations d'influence seront exploitées dans le point suivant afin de définir la structure multidimensionnelle du schéma de l'ED. Nous considérons en effet dans notre méthode que les faits de l'ED sont des concepts centraux sur lesquels *influent* les concepts de dimensions. Les relations d'influence jouent ainsi un rôle important dans la définition du schéma de l'ED, et permettent d'inférer de nouveaux concepts multidimensionnels.

La propagation des relations d'influence repose sur la définition de règles de propagation, que nous avons définies en nous inspirant des travaux sur les graphes causaux qui utilisent la même sémantique des relations d'influence [43]. Dans la littérature portant sur la notion de but, un ensemble de buts est représenté par un graphe de buts, dont les sommets représentent les buts et les arrêtes représentent les relations d'influence entre les buts. La propagation de ces relations d'influence repose sur deux opérations : l'*addition* qui correspond à l'idée de cumul des influences de plusieurs chemins ayant le même but comme destination, et la *multiplication* correspondant à l'idée de transitivité des influences dans un chemin. En nous basant sur le modèle de buts proposé, nous définissons les relations de propagation suivantes (B_1, B_2 et B_3 sont des instances de la classe But) :

– Si B_1 influence positivement B_2, et B_2 influence négativement B_3 alors B_1 influence négativement B_3
– Si B_1 influence négativement B_2, et B_2 influence négativement B_3 alors B_1 influence positivement B_3
– Si B_1 influence positivement B_2, et B_2 influence positivement B_3 alors B_1 influence positivement B_3
– Si B_1 influence négativement B_2, et B_2 influence positivement B_3 alors B_1 influence négativement B_3
– Si B_1 influence positivement B_2, et B_2 influence négativement B_3 alors B_1 influence B_3 (de manière indéteminée).

La formalisation de ces règles en langage $SWRL$[8] sur l'ontologie OED donne les règles suivantes (B_1, B_2 et B_3 représentent les instances des buts *But*, *Positive-influence* et *Negative-influence* représentent les relations (rôles) d'influence négatives et positives entre les buts) :

$Positive-influence(B_1, B_2) \wedge Negative-influence(B_2, B_3) \longrightarrow Negative-influence(B_1, B_3)$

8. http ://www.w3.org/Submission/SWRL/

$Negative - influence(B_1, B_2) \wedge Negative - influence(B_2, B_3) \longrightarrow Positive - influence(B_1, B_3)$

$Positive - influence(B_1, B_2) \wedge Positive - influence(B_2, B_3) \longrightarrow Positive - influence(B_1, B_3)$

$Negative - influence(B_1, B_2) \wedge Positive - influence(B_2, B_3) \longrightarrow Negative - influence(B_1, B_3)$

$Positive - influence(B_1, B_2) \wedge Negative - influence(B_1, B_3) \longrightarrow Undetermined - influence(B_1, B_2)$

Une fois ces relations d'influence propagées, nous enrichissions notre modèle de besoins par ces nouvelles relations d'influence inférées au niveau de la classe *Contribution*.

1.3.4.2.3 Définition du schéma multidimensionnel de l'ED

Le modèle multidimensionnel de l'ED est représenté par l'OED annotée par les concepts multidimensionnels (faits, mesures, hiérarchies de dimensions et attributs de dimension). Un *fait* représente le sujet ou le thème analysé. Un fait est formé de *mesures* ou attributs du fait qui correspondent aux informations liées au thème analysé. Une *dimension* représente un contexte d'analyse d'un fait. Les dimensions se présentent sous forme d'une liste d'éléments organisés de façon *hiérarchique*. Les dimensions sont caractérisées par des *attributs* de dimensions. Nous utilisons la primitive OWL *rdfs :label* pour annoter l'ontologie OED par les concepts multidimensionnels.

L'identification des faits et des dimensions se base d'abord sur les buts des utilisateurs. Ce processus est conforme aux projets réels d'ED, où le modèle multidimensionnel généré doit d'abord répondre aux besoins des décideurs. Selon *Ralph Kimball* [109], les mesures sont des *indicateurs* clés de performances et les dimensions représentent le *contexte* selon lequel les mesures sont analysées. Nous nous basons sur cette définition pour définir les faits et les dimensions. Dans notre modèle de but, les résultats des buts (et leurs métriques) correspondent aux indicateurs à analyser, et les critères des buts correspondent aux contextes d'analyse. Nous considérons que les critères des autres buts influençant le but en cours (par la relation de contribution positive ou négative) sont également des dimensions candidates, puisque ces critères font aussi partie du contexte d'analyse du but.

Nous validons ensuite les liens entre les faits et les dimensions par les relations entre les concepts de l'ontologie locale OED. La littérature relative aux propositions des modèles multidimensionnels, distingue les configurations usuelles qui ne créent aucune erreur d'agrégation (ou summarizability) (cf. section 2.2.2.5). Ces configurations consistent à associer un fait à sa dimension par une relation *un à plusieurs* (relation (0..*) du coté du fait et (1..1) du coté de la dimension), et à associer une relation *un à plusieurs* entre le niveau de dimension inférieur (dl) et son niveau supérieur (ds) dans une hiérarchie de dimension (relation (1..*) du coté de dl et (1..1) du coté de ds). Ces multiplicités sont représentées dans le formalisme des LD par les contraintes de cardinalités sur les rôles ($\leq nP.C$ pour les cardinalités minimales, et $\geq nP.C$ pour la cardinalité maximale) ou par la contrainte de fonctionnalité sur un rôle r ($functional$ r).

Ces multiplicités peuvent être définies selon des relations directes entre les classes, ou selon des relations transitives. L'identification des relations transitives se fait en propageant les multiplicités. Ceci se fait en définissant des règles de propagation transitives et en utilisant un raisonneur pour propager ces règles comme proposé dans [165]. *Romero et al.* étudient dans [165] la complexité de ce mécanisme de raisonnement de propagation des relations un à plusieurs par la

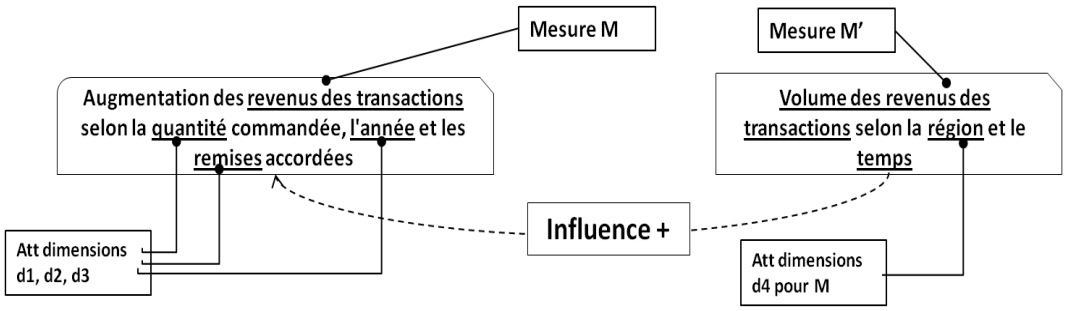

FIGURE 1.26 – Exemple de raisonnement sur les relations d'influence entre les buts

relation de transitivité et démontrent que dans un cas pratique, cette complexité est polynomiale par rapport au nombre de classes pour la plupart des ontologies réelles.

Pour les besoins de l'algorithme, nous nommons Ref_{FD} la fonction qui retourne pour une classe donnée en paramètre, l'ensemble des classes liées à cette classe par une relation respectant la contrainte de cardinalité entre le fait et sa dimension. Nous nommons également Ref_{Hr} la fonction qui retourne pour une classe donnée en paramètre, l'ensemble des classes liées à cette classe par une relation respectant la contrainte de cardinalité entre les niveaux de dimensions.

L'annotation multidimensionnelle de l'ontologie OED se faite selon l'algorithme suivant. Nous considérons ainsi pour chaque instance d'un modèle de but B que :

- les ressources *Métrique* représentent le sujet à analyser et sont donc candidates à représenter une mesure si la ressource est un rôle, ou un fait si la ressource est une classe.
- Les ressources *Critère* représentent le contexte pouvant influencer les mesures et correspondent donc à des attributs de dimension si la ressource est un rôle, ou à une dimension si la ressource est une classe.
- Nous considérons l'ensemble des ressources *Critère* des autres classes But qui influencent ce but précis B comme des candidats potentiels à représenter d'autres dimensions. Si nous considérons l'exemple des deux buts suivants extraits de la figure 1.21) : "*augmentation des recettes des transactions selon la quantité commandée, l'année et les remises accordées*" et "*volume des recettes des transactions selon la région et le temps*". Le deuxième but influence positivement le premier but. Le critère d'analyse *Région* du deuxième but sera candidat à représenter une nouvelle dimension candidate pour la mesure *Recette* du premier but (figure 1.26).
- La classe domaine d'un rôle identifié comme mesure représente un fait.
- La classe domaine d'un rôle identifié comme attribut de dimension représente une dimension.
- Nous validons les liens entre les faits et leurs dimensions si les classes ontologiques correspondantes sont liées par la relation Ref_{FD}.
- Nous validons les liens entre les niveaux de dimensions si les classes ontologiques correspondantes sont liées par la relation Ref_{Hr}.

L'algorithme 1 formalise ces étapes. L'algorithme prend en entrée l'ontologie locale OED formalisée comme suit : $< C, R, Applic(C), Sub(C), Ref(C), Ref'(R), Formalisme, Besoin >$ et l'ensemble des buts. Chaque but est formalisé comme suit : But : <*Metrique, Resultat, In-*

fluence(B)>. *Influence(B)* est la fonction qui associe à chaque but B l'ensemble des buts liés à ce but par une relation de contribution positive ou négative. L'algorithme fournit en sortie l'ontologie locale OED annotée par les concepts multidimensionnels, formalisée comme suit : $< C, R, Applic(C), Sub(C), Ref(C), Ref'(R), Formalisme, Besoin, Annot_F(C), Annot_D(C), Annot_{Mes}(R), Annot_{AttD}(R), Annot_{Hr}(C, C) >$.
$Annot_F(C)$ est la fonction qui annote le concept C par l'annotation 'Fait'. $Annot_D(C)$ est la fonction qui annote le concept C par l'annotation 'Dimension'. $Annot_{Mes}(R)$ est la fonction qui annote le rôle R par l'annotation 'mesure du fait'. $Annot_{AttD}(R)$ est la fonction qui annote le rôle R par l'annotation 'attribut de dimension'. $Annot_{Hr}(C, D)$ est la fonction qui annote les hiérarchies de dimension, où l'ensemble des classes de dimension D présentent la hiérarchie de dimension dont la racine est C. Pour les besoins de l'algorithme, nous introduisons la fonction *Dom(R)* qui est la fonction inverse de la fonction *Applic*, et retourne la classe domaine d'un rôle R.

Le concepteur doit valider les résultats obtenus par l'algorithme d'annotation. Les hiérarchies des classes (relations de subsomption) peuvent être gardées dans le modèle selon le choix du concepteur. Le modèle multidimensionnel obtenu après application de l'algorithme d'annotation *Algorithme1* sur l'ontologie et les besoins du banc d'essai SSB permet d'annoter la classe *Order* comme fait et les classes *Supplier*, *Customer*, *Part* et *Date* comme des dimensions. Des hiérarchies de dimensions sont identifiées comme (*Customer-City-Nation-Region*), (*Supplier-City-Nation-Region*) et (*Part-Category-Brand*). Dans ce scénario, l'ontologie couvre totalement les besoins des utilisateurs ($OED = OP$).

1.3.4.3 La modélisation logique

Le modèle logique relationnel de l'ED est généré à partir de l'ontologie OED annotée, par traduction de ses constructeurs ontologiques en constructeurs relationnels. Plusieurs méthodes proposent de traduire une ontologie décrite selon un formalisme donné (PLIB, RDF, OWL) en un schéma logique, généralement relationnel ou relationnel-objet. Cette traduction peut se faire selon les trois approches [2] : *verticale*, *binaire* et *horizontale*. Nous utilisons les règles de correspondance permettant de traduire une ontologie en un schéma relationnel selon ces différentes représentations qui ont été définies dans le cadre de la thèse [56] effectuée au laboratoire LIAS. Nous rappelons (cf. section 1.2.2.3) qu'une représentation *verticale* représente les données dans une table unique à trois colonnes (sujet, prédicat, objet). Le schéma relationnel traduisant l'ontologie est obtenu par la création d'une table unique de trois colonnes. Chaque ressource est décrite par un ensemble de triplets. La colonne *sujet* représente le nom de la ressource (classe, propriété ou relation). La colonne *prédicat* représente le type décrit par le triplet. La colonne *objet* représente l'objet décrit par la ressource.

Dans une représentation *binaire*, les classes et les propriétés ontologiques sont stockées dans des tables de structures différentes. Une représentation *horizontale* associe à chaque classe de l'ontologie une table ayant une colonne pour chaque propriété de la classe. Nous décrivons les algorithmes utilisés.

1. Approche horizontale
   - Classe : chaque classe correspond à une table. Cette table contient un champ représentant la clé primaire de type entier et un champ pour chaque propriété applicable à cette classe.

- Propriété de type simple (Datatype property) : toute propriété de type simple mono-valuée i.e de cardinalité (0..1 ou 1..1) est représentée comme un champ dans la table correspondant à la classe. Toute propriété collection (multi-valuée) est représentée par une table d'association. Une clé étrangère référençant la table correspondant à la classe définissant la propriété est créée. Un second champ contenant chaque valeur de collection est créé. Chaque type de données XSD supporté par le formalisme OWL possède son type de données SQL correspondant (Exemple. Short devient SMALLINT).
- Propriété de type objet (Object property) : chaque propriété objet mono-valuée est représentée dans la table correspondant à la classe qui la définit (classe domaine) comme une clé étrangère référençant la clé primaire de la table correspondant à la classe référencée (classe rang). Les propriétés objet de type collection présentant une association (1..*) sont représentées dans la table correspondant à la classe référencée par une clé étrangère référençant la clé primaire de la table référençante (classe domaine). Les propriétés objet de type collection présentant une association (n..m) sont représentées par une nouvelle table d'association comportant les champs suivants : une clé étrangère référençant la table correspondant à la classe domaine de la propriété, un champ contenant l'identifiant de l'instance cible et un champ spécifiant la table qui contient cette instance. La clé primaire de la table est constituée des deux premiers champs.

2. Approche binaire
   - Classe : chaque classe correspond à une table contenant un seul champ représentant une clé primaire de type entier.
   - Propriété de type simple : chaque propriété de type simple (monovaluée et multivaluée) est représentée par une table d'association dont les champs sont : une colonne de la clé primaire identifiant les individus de la classe domaine, une deuxième colonne contenant les valeurs de propriété et une troisième colonne de type chaîne de caractère permettant de référencer la table de l'individu (dans la hiérarchie des classes).
   - Propriété de type objet : pour chaque propriété de type objet (monovaluée et multivaluée), une table d'association est crée. Les champs de cette table sont : une colonne pour la clé primaire de la table représentant les identifiants des instances de la classe source (domaine), une colonne représentant les identifiants de la classe cible (co-domaine), deux colonnes de type chaîne de caractères spécifiant la table des instances source et cible.

Les caractéristiques des propriétés (symétrie, transitivité), pour les trois approches s'effectuent par la création de triggers. Les deux *algorithmes* 2 et 3 formalisent les étapes de traduction respectivement pour l'approche horizontale et binaire.

begin
Entrées : Ontologie OED : $< C, R, Applic, Ref(C), Ref(R), Formalisme >$
et ensemble de buts : $< Metrique, Critere, Influence(B) >$
Sorties : Ontologie OED annotée par les concepts multidimensionnels
OED :$< C, R, Applic, Ref(C), Ref(R), Formalisme,$
$Annot_F(C), Annot_D(C), Annot_{Mes}(R), Annot_{AttD}(R), Annot_{Hr}(C, < C >)$
f, *d* : Ressource ;
pour *chaque But B* **faire**
`/*Annotation des faits et leurs mesures à partir des buts*/`
si *Metrique(B) non annotée* **alors**
si *Metrique(B)* $\in R$ **alors**
Annot$_{Mes}$(Metrique(B));
Annot$_F$(Dom(Metrique(B)));
f := Dom(Metrique(B));
fin
sinon si *Metrique(B)* $\in C$ **alors**
Annot$_F$(Metrique(B));
f :=Metrique(B) ;
fin
fin
`/*Annotation des dimensions et leurs attributs à partir des buts*/`
d := Critere(B) ;
si *((d* $\in R$*)* $\wedge$ *(Dom(d)*$\in$ *Ref$_{FD}$(f)))* **alors**
Annot$_{AttD}$(*d*);
Annot$_D$(Dom(*d*));
fin
sinon si *((d* $\in C$*)*$\wedge$ *(d*$\in$ *Ref$_{FD}$(f)))* **alors** Annot$_D$(*d*)
`/*Annotation des dimensions à partir des relations de contribution entre buts*/`
pour *chaque But B' ∈ Influence(B)* **faire**
d := Critere(B') ;
si *((d* $\in R$*)* $\wedge$ *(Dom(d)*$\in$ *Ref$_{FD}$(f)))* **alors**
Annot$_{AttD}$(*d*);
Annot$_D$(Dom(*d*));
fin
sinon si *((d* $\in C$*)*$\wedge$ *(d*$\in$ *Ref$_{FD}$(f)))* **alors** Annot$_D$(*d*)
fin
fin
Rejeter les classes faits sans dimensions
`/*Annotation des hiérarchies de dimensions*/`
pour *chaque classe dimension d* **faire**
Annot$_{Hr}$(d,Ref$_{Hr}$(*d*));
fin
end

Algorithme 1: Annotation multidimensionnelle de l'ontologie OED

begin
Entrées : Ontologie OED
Sorties : Schéma relationnel de l'ontologie selon l'approche horizontale
`/*Traduction des classes*/`
pour *chaque chaque Classe de l'OED* **faire**
La classe devient une table
Associer la clé primaire à cette table (peut être créée automatiquement)
Créer une colonne pour chaque propriété applicable à cette classe
fin
`/*Traduction des rôles de type Attribut*/`
pour *chaque Datatype property (Dp)* **faire**
si *Dp est mono-valué, i.e de cardinalité (0..1 ou 1..1)* **alors**
Dp devient une colonne dans la table correspondante
fin
sinon si *Dp est multi-valué, i.e une collection* **alors**
Dp devient une table d'association portant le nom (NomClasse_NomDp)
Deux colonnes sont crées à la table :
– Une clé étrangère référençant la table correspondant à la classe définissant le Dp est crée ;
– Un champ contenant chacune des valeurs de la collection ;
fin
fin
`/*Traduction des rôles de type Relation*/`
pour *chaque Object property (Op)* **faire**
si *représente une association (1..*)* **alors**
Op est représentée une clé étrangère dans la table rang référençant la clé primaire de la table domaine. Le nom de cette colonne est Nom(Op)_ref ;
Une deuxième colonne nommé Nom(Op)_refTable est ajoutée pour préciser le nom de la table de la classe domaine à laquelle appartient l'instance ;
fin
sinon si *Op représente une association (n..m)* **alors**
Op est représentée par une nouvelle table d'association, nommée Nom(ClasseDomaine)_Nom(Op) qui contient trois champs :
Une clé étrangère référençant la table correspondant à la classe domaine ;
Une colonne nommée Nom(Op)_ref pour l'identifiant de l'instance cible ;
Une colonne nommée Nom(Op)_refTable spécifiant la table qui contient cette instance ;
La clé primaire de la table = (Une clé étrangère + l'identifiant de l'instance cible) ;
fin
fin
end

Algorithme 2: Algorithme de traduction de l'ontologie en modèle relationnel selon l'approche horizontale

begin
Entrées : Ontologie OED
Sorties : Schéma relationnel de l'ontologie selon l'approche binaire
`/*Traduction des classes*/`
pour *chaque Classe de l'ODE* **faire**
La classe devient une table
Associer la clé primaire à cette table (peut être créée automatiquement)
fin
`/*Traduction des rôles de type Attribut*/`
pour *chaque Datatype property Dp (mono-valué ou multi-valué)* **faire**
Dp est représentée par une table d'association nommée NomClasse_NomPropriété contenant les champs suivants :
- Une colonne de la clé primaire, nommée rid identifiant les individus de la classe domaine ;
- Une deuxième colonne contenant les valeurs de propriété ;
- Une troisième colonne, nommée rid_Table de type chaîne de caractère permettant de référencer la table de l'individu ;

fin
`/*Traduction des rôles de type Relation*/`
pour *chaque Object property Op (mono-valué ou multi-valué)* **faire**
Op est représentée par une table d'association nommée NomClasse_NomPropriété contenant les champs suivants :
- Une colonne pour la clé primaire, nommée rid représentant les identifiants des instances de la classe source (domaine) ;
- Une colonne représentant les identifiants de la classe cible (co-domaine) ;
- Deux colonnes de type chaîne de caractères spécifiant la table des instances source et cible.

fin
end

Algorithme 3: Algorithme de traduction de l'ontologie en modèle relationnel selon l'approche binaire

La traduction de l'ontologie OED annotée en un schéma relationnel selon une approche horizontale permet d'obtenir un schéma en flocon de neige. Le fait *Order* et ses quatre dimensions (*Customer*, *Supplier*, *Brand* et *Time*) ont été identifiés. Ces dimensions sont éclatées en plusieurs niveaux comme par exemple les hiérarchies de dimensions (Brand-Category) ou (Customer-City-Nation-Region). Le schéma en étoile est obtenu en regroupant les niveaux de dimensions en une seule table de dimension (figure 1.27). Le schéma obtenu couvre totalement le schéma logique (tables et leurs attributs) du banc d'essai SSB.

1.3.4.4 La modélisation physique

Cette dernière phase décrit l'implémentation effective du modèle de l'ED en utilisant un SGBD. Nous décrivons également le processus de persistance des besoins dans la structure de l'ED (figure 1.28). Pour décrire le processus de persistance des besoins et sa faisabilité, nous avons opté pour une architecture de BDBO de type I (base sémantique *Oracle*) et de type III (base sémantique *OntoDB*) afin de représenter l'ED sémantique.

1.3.4.4.1 Persistance des besoins sur la BDBO d'Oracle

Nous décrivons d'abord l'étude de faisabilité en utilisant la base sémantique d'Oracle. Oracle permet de stocker les ontologies en utilisant une représentation *verticale* où tous les triplets (sujet, prédicat, objet) sont stockés dans la même table. Oracle décompose son schéma en plusieurs tables dont les principales sont : RDF_Link, RDF_values, RDF_Node et RDF_Model_Internal. Les

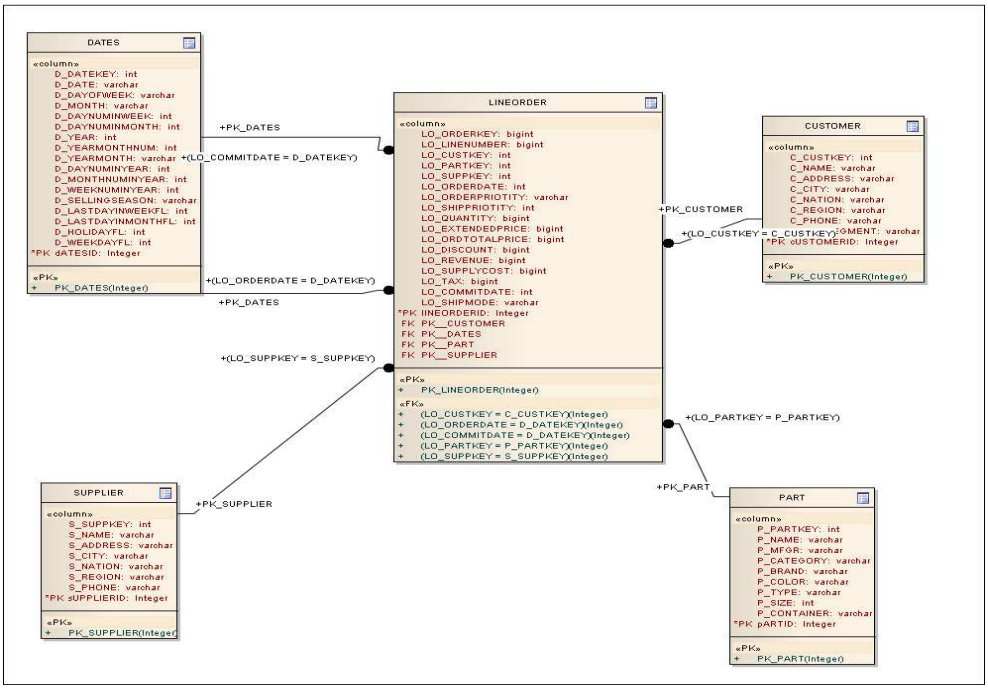

FIGURE 1.27 – Le modèle multidimensionnel relationnel.

données sémantiques sont stockées dans le schéma MDSYS. La BDBO d'Oracle suit une architecture de *type I* où il n'y a pas de séparation entre les différents schémas (ontologiques et données).

La traduction de l'ontologie OED selon une approche verticale peut aussi se faire en utilisant l'éditeur d'ontologies Protégé qui permet de générer un fichier N-triple (sujet, prédicat, objet) à partir d'une ontologie. Nous avons ensuite procédé à la création de l'ED sémantique en créant une nouvelle BDBO Oracle, dont le schéma logique correspond au fichier N-triple obtenu à l'étape précédente. Nous avons utilisé SQL*Loader pour le chargement du fichier N-triple dans la nouvelle BDBO. SQL*Loader est un utilitaire d'Oracle qui permet le chargement de données à partir d'un fichier plat vers une table de la base de données Oracle.

Comme la démarche le préconise, nous étendons le métaschéma ontologique d'Oracle par le modèle des besoins défini. Nous avons donc représenté les entités du modèle de besoins par des classes, les attributs du modèle par des data-value Property, les relations entre les entités par des Object-Property et les données par des instances. Le métaschéma ontologique d'Oracle suit également une représentation verticale (sujet, prédicat, objet). Le modèle de besoins l'étendant doit se conformer à cette représentation logique. Les nouvelles *classes* (But, Résultat, Critère, etc), les *data-value properties* associées à la classe But (nom, priorité, contexté, etc) et les *object properties* entre les classes (ResponsableDe(Actor, Goal), aCritère(Goal, Criteria), etc) du modèle de besoins sont créées pour enrichir le métaschéma. L'extension du schéma ontologique du modèle d'Oracle par le modèle des besoins est illustrée dans la figure 1.29.

Le modèle de besoins sera instancié sous forme de triplets par l'ensemble des buts spécifiés. Par exemple, en reprenant l'exemple du premier but SSB, ce besoin sera représenté par un ensemble de triplets comme (*but1*, *aCritere*, *Produit*), (*but1*, *aResultat*, *Augmenter recette*), etc.

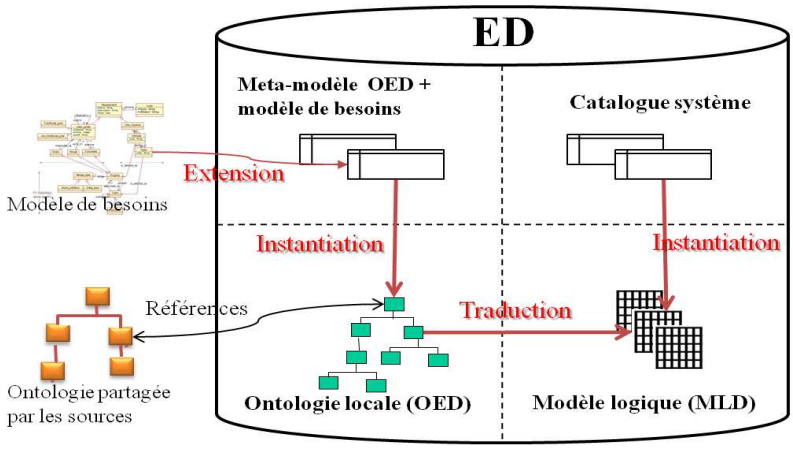

FIGURE 1.28 – ED sémantique et persistance des besoins des utilisateurs.

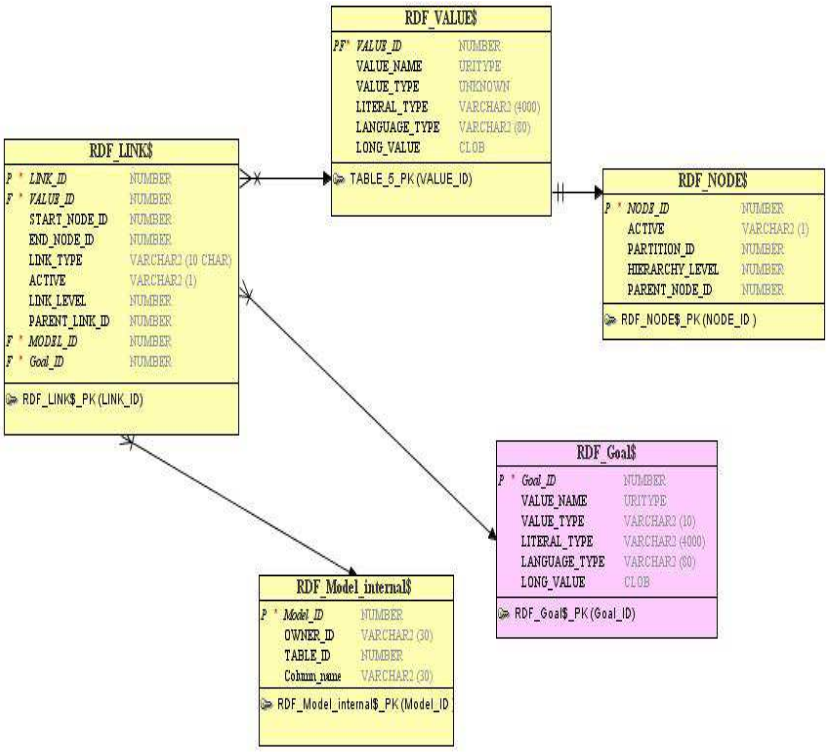

FIGURE 1.29 – Extension du métaschéma sémantique d'Oracle par le modèle de besoins

L'ensemble des triplets sont sauvegardés au niveau de la table RDF_Goal créée. Ces besoins pourront être restitués à tout moment en interrogeant l'ED par des requêtes SQL ou SPARQL sur la table RDF_Goal.

1.3.4.4.2 Persistance des besoins sur la BDBO OntoDB

Nous avons également validé notre démarche de persistance des besoins dans la BDBO OntoDB possédant une architecture de *type III*. La présence du métaschéma dans OntoDB offre une plus grande flexibilité et permet son extension par le modèle des besoins. De la même manière, nous avons utilisé l'ontologie OED obtenue à partir des buts spécifiés sur l'ontologie SSB. Cette ontologie est ensuite traduite en schéma relationnel horizontal, qui est la représentation logique adoptée par OntoDB. Nous avons créé une nouvelle BDBO OntoDB dont la structure est constituée du : modèle logique (relationnel horizontal) de données, de son méta-modèle, du schéma de l'ontologie locale OED et du méta-schéma de l'ontologie étendu par le modèle de besoins. Cette extension se fait selon un schéma horizontal où chaque classe du modèle des besoins est représentée par une table Entity dans le métaschéma d'OntoDB. Cette extension se fait en utilisant le langage ontologique défini pour OntoDB, qui est le langage *OntoQL*. Par exemple, les requêtes OntoQL suivantes permettent la création du modèle de besoins dans le métaschéma :

```
Create Entity #GoalModel(#Property (#Domain String, #Name String, #Date Date), #collects REF(#Goal))

Create Entity #Result(#its_properties REF (#Property))

Create Entity #Metric(#its_properties REF (#Property))

Create Entity #Criteria (#its_properties REF (#Property) Array)

Create Entity #Goal (#Property (#Id Int, #Name String, #Description String, #Context String,

#Purpose String, #Priority String), #Achieves REF (#Result),

#Measured_by REF (#Metric), #HasCriteria REF (#Criteria), #ResponsibleOf REF(#Actor))

Create Entity Functional_Goal UNDER #Requirement

Create Entity NonFunctional_Goal UNDER #Requirement
```

La restitution des besoins à partir de cet ED sémantique se fait également par des requêtes *OntoQL*. Par exemple la requête suivante permet l'extraction des mesures des buts :

```
Select g.#hasResult from #Goal g where g.#oid = " + resultSet.getInt(1));
```

Nous avons montré ci-dessus la faisabilité de notre approche sur des architectures de BDBO afin de représenter le modèle logique de données, le modèle conceptuel de données (l'ontologie locale) et le modèle de besoins. Nous précisons qu'il est possible de déployer l'ED sur une architecture de BD classique (Oracle par exemple) en effectuant le même processus de création des tables du modèle relationnel obtenu et en étendant le méta-schéma de la BD créée.

1.4 Validation : simulation de l'optimisation de l'ED

Nous montrons dans cette section l'intérêt de la persistance des besoins dans la structure d'un ED par un ensemble d'expérimentations simulant la tâche d'optimisation.

1.4.1 Motivations de l'expérimentation

Les expérimentations menées ont pour but de montrer que l'utilisation des besoins va au-delà des phases de définitions des besoins et de modélisation conceptuelle. Nous montrons que les besoins des utilisateurs peuvent être utilisés pour simuler différentes tâches du cycle de vie de l'ED comme la tâche d'optimisation. La diversité des modèles de données et des architectures de déploiements du cycle de conception des ED rend nécessaire cette simulation dès les premières phases de conception. Nous menons des expérimentations afin de comparer le résultat et le coût d'une optimisation dirigée par les besoins (obtenus dès le début de la conception de l'ED), et d'une optimisation dirigée par les requêtes des utilisateurs (obtenues après la conception de l'ED et son exploitation par les utilisateurs). Cette comparaison nous permet de montrer si une simulation de cette tâche de conception est possible à partir des besoins des utilisateurs. Nous rappelons la règle 80-20 énoncée par *R.Elmasri* dans [55] qui s'applique à ce contexte de simulation : 80% de la charge de travail utilisée lors la conception physique de la base de données, est représentée par 20% des transactions identifiées. Les expérimentations que nous menons ont trois principaux objectifs :

1. Vérifier si une simulation des phases de conception de l'ED à partir des besoins des utilisateurs est possible.
2. Renforcer les liens entre les différentes phases de conception de l'ED afin de faciliter la communication entre les acteurs (administrateur, concepteur, éditeur, etc) intervenant tout au long du cycle de conception.
3. Apporter un nouveau modèle économique dans la conception des ED. En effet, nous avons remarqué que dans le contexte actuel, le rôle du concepteur physique diminue au point d'être parfois remplacé par des outils d'administration automatiques. Cependant, la robustesse de ces outils est souvent remise en cause [54]. Avec ce nouveau modèle illustré en figure 1.30, il est préférable d'exploiter la ressource humaine encore disponible *le concepteur* en le faisant interagir avec le système sur les différents niveaux de conception de l'ED. Nous tentons ici de fournir ce lien "humain" entre la conception logique et la conception physique.

Nous présentons dans ce qui suit l'intérêt de la persistance des besoins pour la tâche d'optimisation.

1.4.2 Persistance des besoins pour l'optimisation de l'ED

1.4.2.1 Présentation de l'expérimentation

Les différentes tâches d'optimisation d'un ED s'effectuent habituellement au niveau physique et reposent sur les principales entrées suivantes : le *schéma* de l'ED, une *charge de requêtes fréquentes*, un ensemble de *contraintes* données (espace de stockage, coût de mise à jour, etc), les *caractéristiques du SGBD* et l'*architecture de déploiement* (centralisé, distribué, etc). L'ensemble

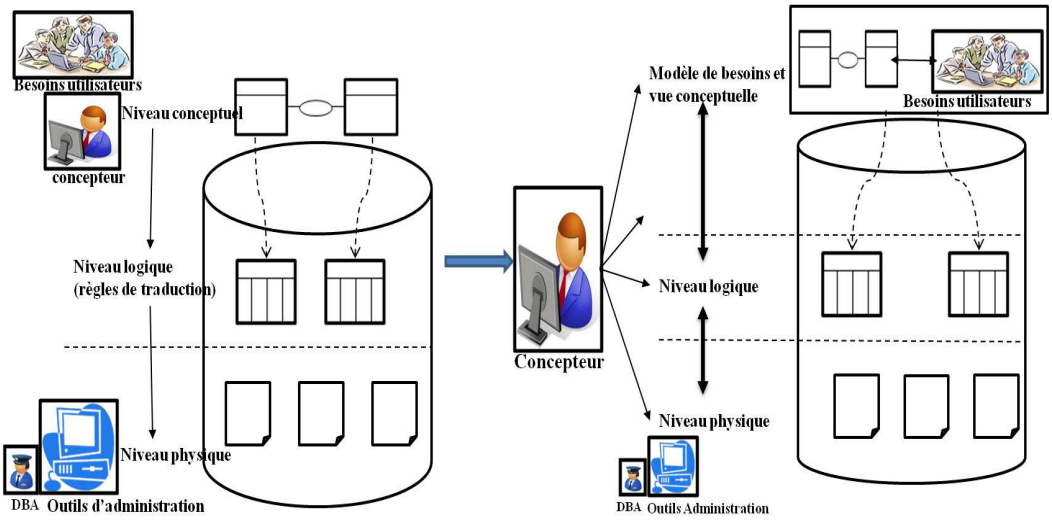

FIGURE 1.30 – Rôles des acteurs dans l'architecture ANSI/SPARC

de ces entrées peut être fourni après l'analyse des besoins des utilisateurs. Toutes les contraintes (comme l'espace de stockage et le coût de mise à jour) et les caractéristiques du SGBD sont collectées auprès des acteurs du système sous forme de besoins non fonctionnels. Le schéma de l'ED est défini à partir des entités et des propriétés identifiées après analyse des besoins. La charge de requêtes correspond aux traitements que devra assurer l'ED, qui sont également obtenus à partir des besoins des utilisateurs. Nous rappelons que cette charge de requêtes n'est habituellement obtenue qu'une fois l'ED spécifié au niveau physique ensuite exploitée par les utilisateurs finaux de l'ED pendant une certaine période de temps. Durant cette période sensible où les statistiques et requêtes fréquentes de l'ED ne sont pas disponibles, nous proposons d'exploiter les besoins des utilisateurs pour simuler le processus d'optimisation.

Nous proposons deux cas d'étude : le problème de *sélection des index* et celui de la *fragmentation horizontale* et nous reformulons ces problèmes en prenant comme entrée les besoins des utilisateurs.

1.4.2.2 Persistance des besoins pour la définition des index

Nous commençons par formaliser le problème de *sélection d'index* en exploitant les besoins des utilisateurs définis sous forme de buts. Les index représentent une technique d'optimisation de requêtes très efficace et très répandue dans le contexte des ED. Ils permettent de réduire le coût d'exécution des requêtes en minimisant le volume de données à exploiter dans les calculs. Un problème classique de sélection d'index se formalise comme suit. Etant donné :

- I = $I_1, ..., I_n$ un ensemble d'index candidats (obtenus à partir des prédicats de sélection des **requêtes**).
- Q = $Q_1, ..., Q_m$ un ensemble de **requêtes**.
- S la taille de l'espace de stockage allouée pour les index.

Il faut alors trouver une configuration d'index Config telle que :

- Le coût d'exécution des requêtes soit minimal.
- L'espace alloué pour le stockage de ces index ne dépasse pas S.

Nous adaptons ce problème en exploitant les besoins des utilisateurs au lieu des requêtes des utilisateurs de la manière suivante (figure 1.31) :

- $I = I_1, ..., I_n$ un ensemble d'index candidats (obtenus à partir des **besoins**).
- $B = B_1, ..., B_m$ un ensemble de **besoins**.
- S la taille de l'espace de stockage allouée pour les index.

Il faut alors trouver une configuration d'index Config telle que :

- Le coût d'exécution des requêtes soit minimal.
- L'espace alloué pour le stockage de ces index ne dépasse pas S.

Plusieurs travaux ont proposé des algorithmes pour répondre au problème de sélection d'index. Nous optons pour l'algorithme défini par *Bouchakri et al.* dans [27], proposé dans les cadre des travaux menés au laboratoire LIAS. Pour faciliter l'exploitation des besoins des utilisateurs par le modèle de coût utilisé par l'algorithme, nous traduisons les besoins spécifiés en requêtes SQL. Rappelons qu'une requête SQL correspond à la définition d'un but décrit selon la syntaxe SQL. La traduction des besoins se fait en deux étapes :

- Structuration des besoins décrits en langue naturelle dans la spécification du banc d'essai SSB selon le modèle de but que nous avons défini.
- Traduction des buts obtenus en requêtes SQL. Cette traduction correspond à une transformation *Model To Text* dans l'ingénierie dirigée par les modèles (IDM). Cette étape est réalisée à l'aide de modèles de génération fournies par le plugin *Accelio* disponible sous l'environnement *Eclipse*. Accelio fournit un générateur de code permettant de transformer des modèles vers du code respectant une syntaxe définie, dans le cadre d'une Architecture Dirigée par les Modèles (MDA). Ce processus repose sur la définition d'un algorithme précisant les règles de transformation définies ci-dessous appliquées sur chaque but :

```
SELECT (les propriétés des classes "Métrique" dans le modèle de but) As (Résultat)
FROM (l'ensemble des classes spécifiant l'instance du but)
WHERE (les propriétés des classes "Critère" dans le modèle de but et leurs valeurs.
Les jointures entre les tables Fait et Dimension utilisées sont rajoutées)
```

Les expérimentations sont effectuées sur le schéma logique obtenu. L'ensemble des besoins décisionnels (13 besoins) spécifiés dans la spécification du banc d'essai SSB sont exploités. Les expérimentations ont donné les résultats résumés dans le tableau 1.2.

Ce tableau présente les index générés et le taux d'optimisation du coût d'exécution de chaque besoin. Ces résultats sont confrontés aux index et coût d'optimisation obtenus à partir de la charge de requêtes finales (fournies au niveau de la spécification du banc d'essai SSB). Ces résultats confirment que les index proposés par les besoins couvrent les index proposés par les requêtes du banc d'essai. Les coûts d'optimisation sont également assez similaires. Ceci s'explique par le fait que les requêtes issues des besoins sont similaires aux requêtes du banc d'essai (sans les clauses group by et order by que nous n'avons pas pu générer).

Stockage (GO)	Besoins		Requêtes	
	Index	Coût (%)	Index	Coût (%)
1	d_year	46.7	d_year	46.7
1.5	d_year, s_region	72.7	d_year, s_region	72.7
2	d_year, s_region, p_category	75.1	d_year, s_region, p_category	75.1
2.5	d_year, d_yearmonth, s_region	87.9	d_year, d_yearmonth, s_region	87.9
3	d_year, d_yearmonth, s_région	87.9	d_year, d_yearmonth, s_région	87.9
3.5	d_year, d_yearmonth, s_region, p_category	90.2	d_year, d_yearmonth, s_region, p_category	90.2
4	d_year, d_yearmonth, c_region, s_region, p_category	91.7	d_year, d_yearmonth, c_region, s_region, p_category	91.7
4.5	d_year, d_yearmonth, c_region, s_region, p_mfgr, p_category	92	d_year, d_yearmonth, c_region, s_region, p_mfgr, p_category	91.9
5	d_year, d_yearmonth, c_region, s_region, s_nation, p_category	92.7	d_year, d_yearmonth, c_region, s_region, s_nation, p_category	92.6

TABLE 1.2 – Index générés et taux d'optimisation du coût d'exécution des besoins Vs des requêtes

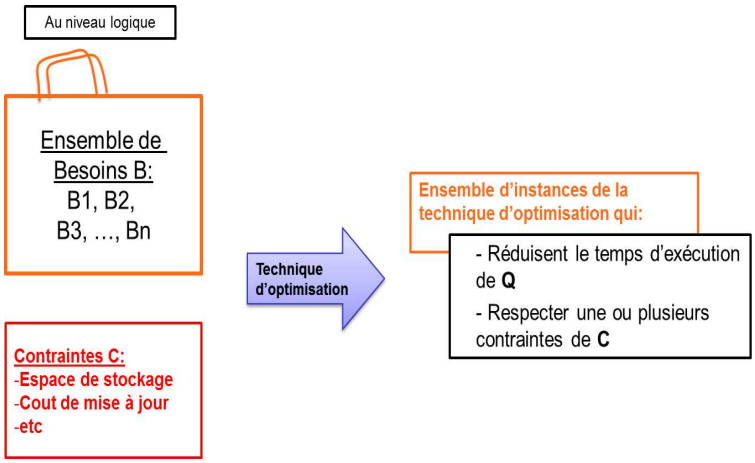

FIGURE 1.31 – Sélection des index de l'ED à partir des besoins

1.4.2.3 Persistance des besoins pour la fragmentation horizontale

Nous avons effectué une expérimentation similaire pour simuler le processus de *fragmentation horizontale* (FH) en utilisant les besoins des utilisateurs. La FH est une technique d'optimisation qui consiste à répartir les tuples d'une table en plusieurs sous ensembles disjoints appelés fragments horizontaux. On distingue deux types de FH : (1) FH primaire définie sur une table de dimension en fonction de ses propres attributs, et (2) FH dérivée définie sur la table des faits en fonction des dimensions fragmentées.

Le problème de FH prend en entrée un ensemble de dimensions et une table de fait, une charge de requêtes et une contrainte de maintenance. Il retourne en sortie un schéma de partitionnement pour l'ensemble des tables qui réduit au mieux le coût de la charge. Le problème de FH est formalisé comme suit [15] :

Ayant les entrées suivantes :

- Un ensemble de table de dimension D=D1,D2,..Dd et une table de fait
- Une charge de requête Q=Q1,Q2,..Qm
- W = seuil de fragments (fixé par l'administrateur)

l'algorithme de FH doit alors fournir en sortie :

- Un ensemble D' inclus ou égal à D des tables de dimension fragmentées
- Un ensemble de N fragments de faits F1,F2,..Fn

Avec comme objectif de :

- Réduire le temps de réponse de Q
- N<=W

Nous adaptons le problème de FH en remplaçant la charge de requêtes par l'ensemble des besoins. Ainsi la formalisation devient comme suit : ayant les entrées suivantes :

- Un ensemble de table de dimension D=D1,D2,..Dd et une table de fait
- Un ensemble de besoin B=B1,B2,..Bm
- W = seuil de fragments (fixé par l'administrateur)

l'algorithme de FH doit alors fournir en sortie :

- Un ensemble D' inclus ou égal D des tables de dimension fragmentées
- Un ensemble de N fragments de faits F1,F2,..Fn

Avec comme objectif de :

- Réduire le temps de réponse de Q
- N<=W

Pour répondre à ce problème, nous utilisons l'algorithme de résolution de FH défini dans [15], qui a été proposé dans le cadre des travaux menés au laboratoire LIAS. Nous avons comparé les résultats de l'optimisation effectuée en utilisant la charge de requêtes réelles du banc d'essai SSB et les requêtes que nous avons traduits à partir des besoins. Les résultats de ces expérimentations sont illustrés dans la figure 1.32 et montrent que les besoins apportent un gain en performance supérieur de 15% comparé aux requêtes. Cette amélioration s'explique par l'absence des clauses Group By. Ces résultats sont prometteurs dans le sens où nous montrons que les besoins permettent de fournir une simulation de l'optimisation par FH pratiquement équivalente à une FH effectuée lors de la phase physique.

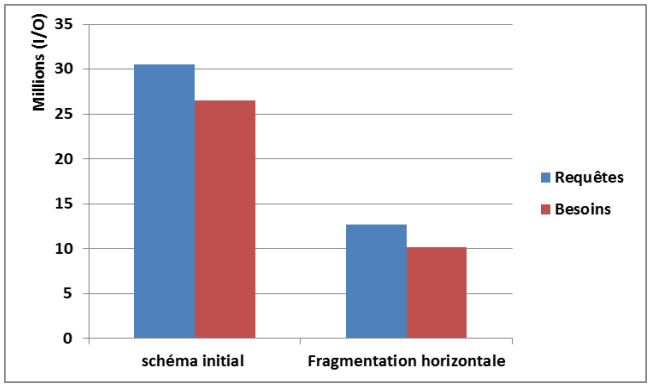

FIGURE 1.32 – Fragmentation horizontale de l'ED à partir des besoins

1.5 Conclusion

Après avoir montré la convergence des méthodes de conception d'ED actuelles vers des approches dirigées par les besoins et où la sémantique du modèle est de plus en plus considérée ; nous avons proposé dans ce chapitre une méthode de conception ontologique du schéma de l'ED permettant la persistance des besoins des utilisateurs dans une structure d'ED sémantique. Nos contributions dans ce chapitre concrétisent le premier objectif énoncé dans l'introduction générale, qui est de consolider le cycle de conception de l'ED par une représentation persistante des besoins.

Pour ce faire, nous avons fourni un modèle de besoins défini selon une approche orientée but, les étapes de la méthode de conception du schéma de l'ED (conceptuel, logique et physique), et le mécanisme permettant de faire persister les besoins dans la structure de l'ED. La méthode proposée prend en entrée une ontologie de domaine partagée par les sources de données. Les besoins collectés auprès des utilisateurs sont projetés sur l'ontologie afin d'identifier les concepts et propriétés pertinentes à représenter dans l'ED, et l'ensemble des traitements que devra effectuer l'ED. Une ontologie locale à l'ED est ainsi définie par extraction de toutes les classes et propriétés intervenant dans la définition des besoins. Cette ontologie est annotée par les concepts multidimensionnels. Les mécanismes de raisonnement de l'ontologie sont exploités afin de valider le modèle de l'ED, et d'inférer de nouvelles structures multidimensionnelles. Le modèle de l'ontologie OED est ensuite traduit en un modèle relationnel selon différentes représentations (horizontale, binaire et verticale) par l'application d'un ensemble de règles permettant la traduction des constructeurs de l'ontologie vers les constructeurs d'un modèle relationnel. Nous avons validé notre proposition par l'implémentation du schéma de l'ED dans une structure de BDBO de type I (*Oracle*) et de type III (*OntoDB*), dont les métaschémas ontologiques ont été étendus par le modèle des besoins. Nous avons également démontré de manière informelle ensuite via des expérimentations l'impact de la persistance des besoins sur le cycle de vie de l'ED. Les expérimentations conduites ont montré l'intérêt de la persistance des besoins sur la tâche d'optimisation de l'entrepôt, plus particulièrement sur les problèmes de sélection d'index et de fragmentation horizontale.

Le prochain chapitre porte sur notre deuxième proposition, qui consiste à consolider le cycle de conception de l'ED en revisitant l'ordre d'exécution de la phase ETL. Nous complétons notre

méthode de conception ontologique présentée dans ce chapitre, pour une approche permettant la conception du schéma de l'ED et son alimentation par les données issues des sources.

Chapitre 2

Vers un système d'ED déployé "à la carte"

Sommaire

2.1 Introduction

Le cycle de conception d'un ED passe par les étapes suivantes : l'analyse de besoins, la modélisation conceptuelle, la modélisation logique, la phase d'extraction-transformation-chargement (ETL) et une phase de modélisation physique [68, 69]. Dans ce cycle de conception, la phase ETL a été introduite après la phase de modélisation logique. Cette dernière inclut des choix d'implémentation, qui influencent le schéma de l'ED et lui imposent un déploiement unique (nous faisons référence au déploiement logique du modèle de l'ED selon le modèle du SGBD cible). Le schéma de déploiement suit généralement les schémas des sources, il est donc figé dès le début du processus de conception. L'analyse de l'évolution du cycle de conception des ED nous a montré la diversification des modèles logiques et physiques et des plateformes de déploiement disponibles. Dans le but d'offrir une méthode de conception plus adaptée au cycle de conception actuel, nous estimons que l'ordonnancement de la phase de conception ETL doit être revu et étudié dès le niveau conceptuel. Le déploiement logique de l'ED pourra ainsi se faire selon plusieurs modèles et architectures en fonction des besoins des utilisateurs et concepteurs.

La définition du modèle de l'ED repose sur une première composante essentielle représentée par les besoins des utilisateurs. Notre première proposition a été de mettre en valeur cette ressource durant tout le cycle de vie de l'ED. Le premier chapitre de la partie *propositions* de ce manuscrit a porté sur la proposition d'une méthode ontologique pour la conception du schéma de l'ED, permettant une représentation persistante des besoins dans la structure de l'ED. La proposition de ce second chapitre exploite la deuxième composante de conception des ED représentée par les "sources de données". La revue de littérature (cf. chapitre 2) nous a permis de constater une convergence des méthodes de conception récentes vers des approches considérant les sources de données au niveau sémantique. Les ontologies sont d'abord apparues dans le domaine de conception des ED, comme une solution facilitant le processus d'intégration des données issues des sources. Les ED peuvent en effet, être considérés comme des systèmes d'intégration de données (SID) matérialisant les données des sources via un processus ETL. Un SID est défini par le triplet <schéma Global (G), Sources (S), Mappings (M)> [120] où G représente le *schéma global* du système d'intégration, S représente l'ensemble des *sources* à intégrer et M décrit différentes *assertions de mapping* entre le schéma global et les schémas des sources. Les ontologies ont souvent été utilisées dans les SID pour fournir une sémantique aux données et faciliter ainsi leur intégration. Certains SID utilisent une ontologie de domaine pour représenter le schéma global G. D'autres SID ont suivi décrivant également les sources par des ontologies locales afin de faciliter les mappings. Tous ces systèmes supposent la définition de sources de données référençant une ou plusieurs ontologies. Ces données référençant des ontologies sont appelées "données ontologiques". Certaines architectures de BD sont définies de manière à pouvoir gérer les données ontologiques ainsi les ontologies que ces données référencent : ce sont les *BDBO*. Les méthodes de conception des ED utilisant des ontologies s'inspirent des travaux sur les SID, afin de faciliter l'intégration des données selon un processus ETL [166]. Cependant, aucune étude ne considère la phase ETL à un niveau complètement conceptuel et aucune proposition n'a été faite pour la définition d'un ED à partir de BDBO. Nous positionnons notre étude dans la lignée de ces travaux et nous exploitons la sémantique du domaine pour la conception et l'alimentation du schéma de l'ED via un processus ETL défini dès la phase conceptuelle, plus précisément au niveau ontologique (figure 2.1). Nous fournissons pour ce faire, un Framework

d'intégration <G,S,M> basé sur le formalisme des logiques de description, et un algorithme ETL basé sur des opérateurs d'intégration conceptuels.

Le deuxième constat important qui ressort de notre analyse de l'état de l'art, est que la plupart des approches proposées étudient séparément les différentes étapes du cycle de conception de l'ED en traitant la modélisation multidimensionnelle indépendamment des problématiques d'intégration et de chargement des données (processus ETL). La méthode que nous proposons complète la méthode de conception du schéma de l'ED (présentée au chapitre précédent) et couvre toutes les phases du cycle de conception.

Par ailleurs, nous rappelons que les BDBO deviennent actuellement des sources potentielles importantes dans les projets d'entreposage. Les BDBO représentent d'excellentes candidates pour une intégration complètement conceptuelle. Pour valider nos contributions, nous proposons de dérouler la méthode de conception proposée en instanciant le Framework d'intégration <G,S,M> par des sources de type BDBO.

Nos contributions pour ce chapitre se résument en les points suivants :

- Proposition d'un Framework d'intégration générique <G,S,M> pour l'intégration des sources de données au niveau conceptuel [100].
- Proposition d'une méthode de conception d'un ED définissant la phase ETL au niveau conceptuel ontologique.
- Application de la méthode de conception proposée par l'instanciation du Framework d'intégration par des sources de type BDBO [23, 18]. Nous fournissons ainsi la première méthode permettant la conception d'un ED à partir d'un ensemble de BDBO.
- Prototypage et tests : un prototype d'outil implémentant l'ensemble des étapes de la méthode est fourni [18]. Des expérimentations ont été conduites, utilisant le banc d'essai ontologique LUBM et permettant d'évaluer les performances de la méthode proposée [22].

Une partie de cette étude, à savoir la proposition de l'algorithme ETL et une partie de l'outil proposé ont été effectuées dans le cadre du magistère de *Berkani Nabila*, étudiante à l'école nationale supérieure d'informatique (ESI), encadré par *Pr.Ladjel Bellatreche* et *Khouri Selma*. Ce chapitre comporte les sections suivantes. La deuxième section présente les fondements théoriques nécessaires à la compréhension de notre proposition. Nous présentons une classification des systèmes d'intégration selon trois critères, que nous enrichissons par deux critères supplémentaires qui sont à l'origine de notre proposition. La troisième section présente nos contributions, à savoir le Framework d'intégration conceptuel et la méthode de conception décrite selon les étapes citées. La quatrième section présente la validation de notre proposition. Nous présentons dans cette section l'instanciation du Framework d'intégration par des sources de type BDBO. Nous prenons l'exemple d'un ensemble de BDBO Oracle. Nous évaluons notre méthode en utilisant le banc d'essai LUBM modélisant le domaine universitaire. Nous présentons un prototype d'outil implémentant toutes les étapes de la méthode proposée. La dernière section conclut le chapitre et récapitule les principaux résultats.

2.2 Fondements théoriques

Nous présentons dans ce qui suit la classification des approches d'intégration proposée par *Bellatreche et al.* dans [16] qui classifie ces approches selon les trois critères orthogonaux suivants :

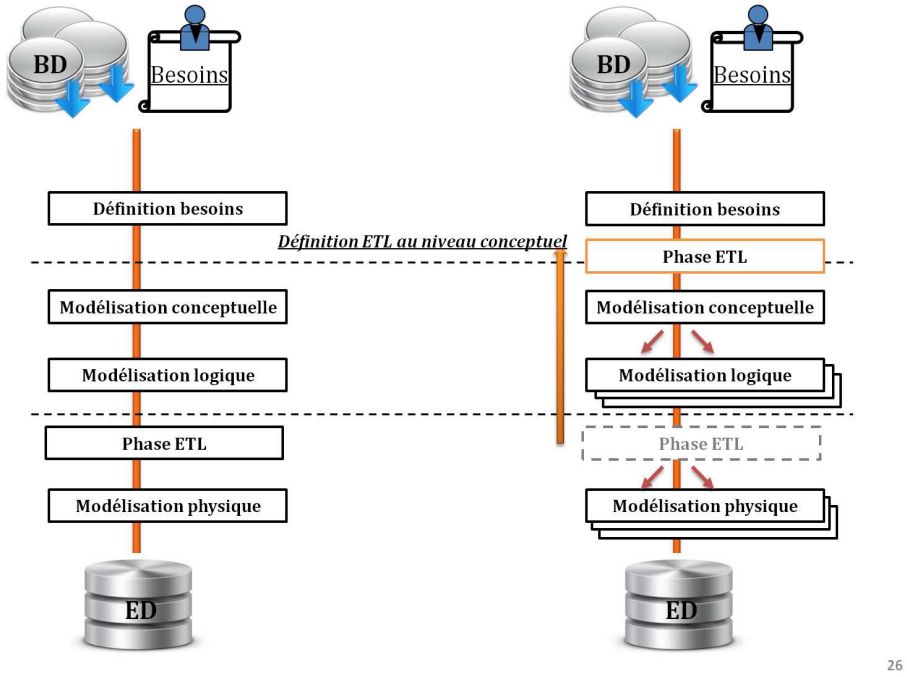

FIGURE 2.1 – Proposition 2 : ETL conceptuel pour un déploiement à la carte

(1) la *représentation des données intégrées*, (2) le *sens de la mise en correspondance* entre le schéma global et les schémas locaux, et (3) le *degré d'automatisation du processus d'intégration*. Nous enrichissons cette classification par deux critères supplémentaires qui sont à l'origine de notre proposition pour ce chapitre (figure 2.2) : le *type de déploiement* et le *niveau d'abstraction* de l'approche d'intégration.

2.2.1 La représentation des données intégrées

Ce critère permet de distinguer les approches d'intégration selon la manière dont elles stockent les données. On distingue : l'approche *virtuelle* et l'approche *matérialisée*.

2.2.1.1 Approche virtuelle

L'approche *virtuelle* (figure 2.3) stocke les données uniquement au niveau des sources. Les SID construits selon cette approche, appelés systèmes médiateurs, reposent sur deux composants : le médiateur dont le rôle est de localiser les données pertinentes à une requête d'utilisateur, et l'adaptateur dont le rôle est d'accéder au contenu de ces sources.

Une requête d'utilisateur est formulée selon le schéma global du médiateur. Des vues abstraites décrivent le contenu de chaque source dans les termes du médiateur. Le médiateur sélectionne les sources pertinentes pour la requête en utilisant ces vues. La requête doit être reformulée afin qu'elle puisse être évaluée sur les sources pertinentes (c'est ce qu'on appelle la réécriture des requêtes). Le résultat de cette réécriture est un ensemble de sous requêtes posées sur les sources. Les adaptateurs, contenant les correspondances entre les données du schéma global et celles des sources, traduisent les sous requêtes dans le langage spécifique de chaque source. Plusieurs SID

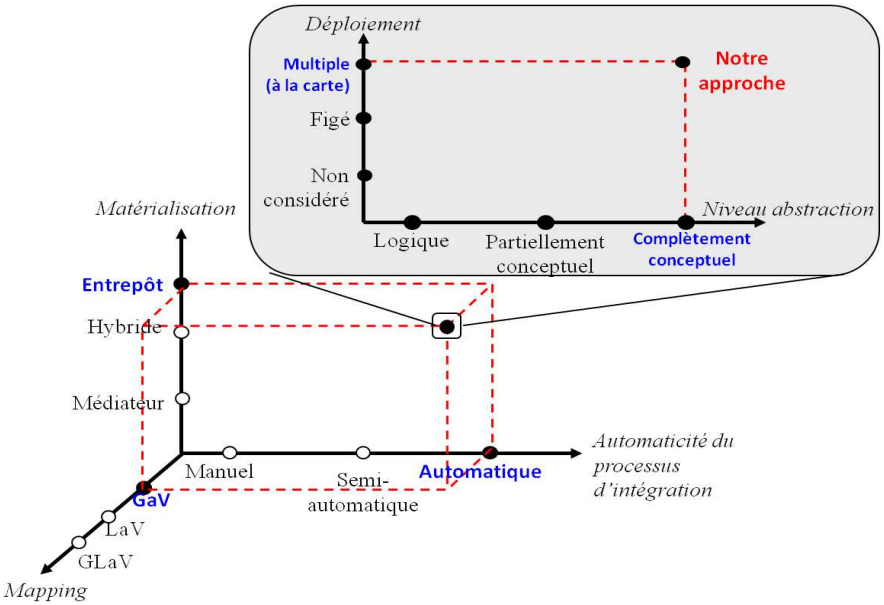

FIGURE 2.2 – Classification des approches d'intégration

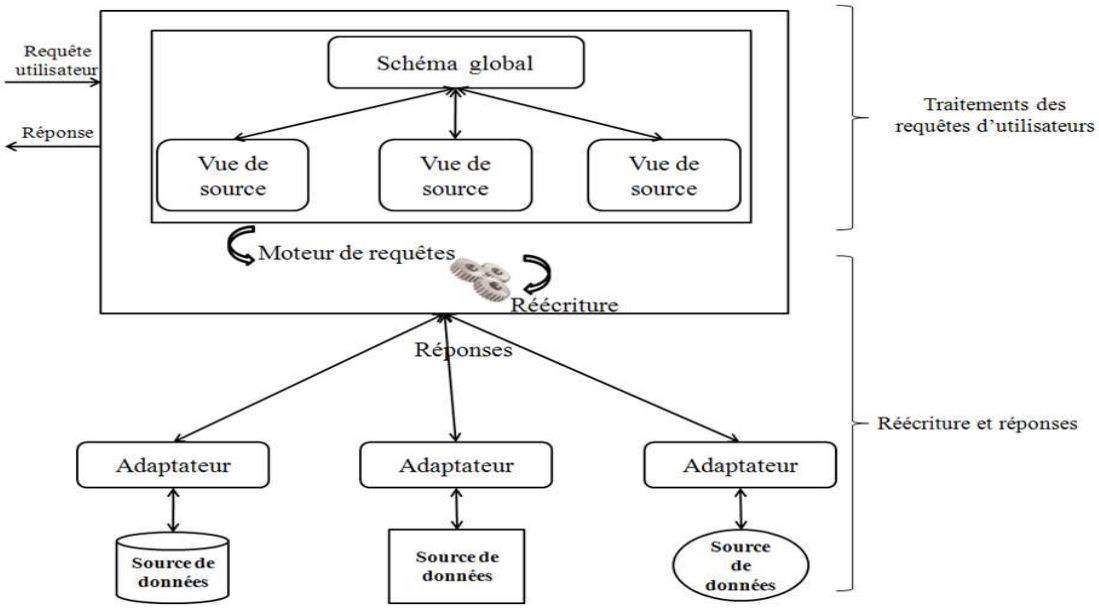

FIGURE 2.3 – Intégration virtuelle [73]

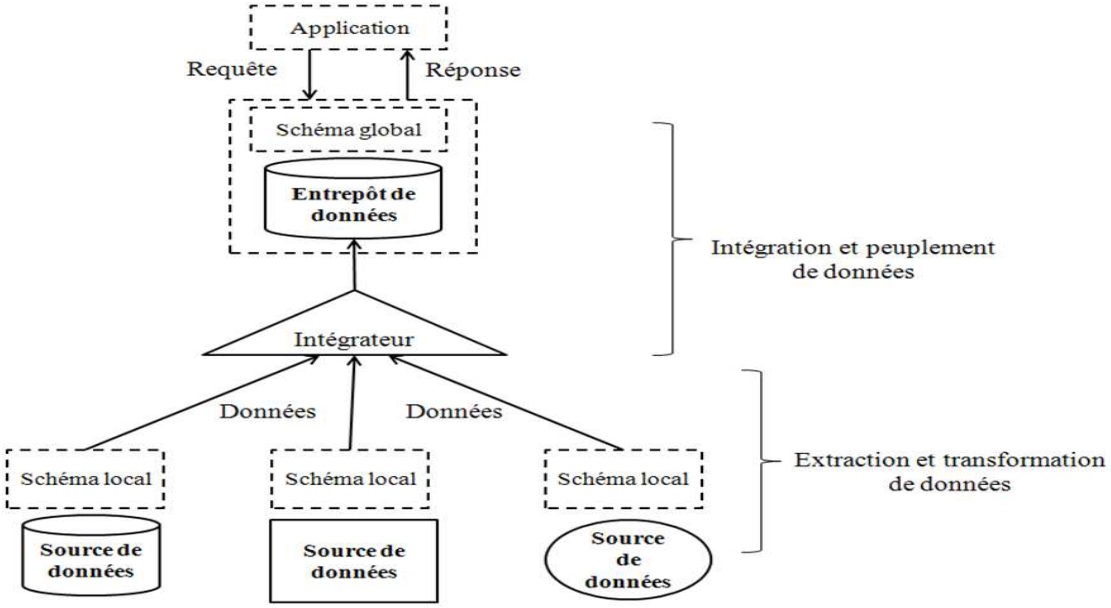

FIGURE 2.4 – Intégration matérialisée [16]

ont été développés selon cette approche comme : le projet *Tsimmis* [44] et le projet *Picsel* [73].

2.2.1.2 Approche matérialisée

L'approche *matérialisée* (figure 2.4) transfère les données des sources au niveau d'un entrepôt de données, il y a donc duplication des données. L'interrogation des données se fait directement sur les données de l'entrepôt et non sur les sources d'origine. Partant de la formalisation <G,S,M> d'un système d'intégration, un ED est dans ce cas vu comme un système d'intégration de données, matérialisant les données via un processus ETL. Le schéma de l'ED peut être le même que le schéma G ou un schéma dérivé de G. Certaines entreprises, pour des raisons de sécurité, préfèrent opter pour une solution d'ED, même si cela implique un coût de stockage et de maintenance des données. Plusieurs SID issus de projets importants ont été développés selon l'approche matérialisée comme le projet américain *Whips* [81], le projet européen *DWQ* (Data Warehouse Quality) [93] et le projet français *OntoDawa* [16].

2.2.2 Le sens de mise en correspondance entre schémas global et locaux

Ce critère permet de distinguer les approches d'intégration selon la liaison entre le schéma global et les schémas locaux des sources à intégrer. On distingue pour ce critère deux approches (figure 2.5) : l'approche *GaV* (Global as View) et l'approche *LaV* (Local as View).

2.2.2.1 Approche GaV

L'approche *GaV* définit le schéma global en fonction des schémas locaux des sources. Chaque schéma (local ou global) est défini comme étant un ensemble de relations. Les relations globales

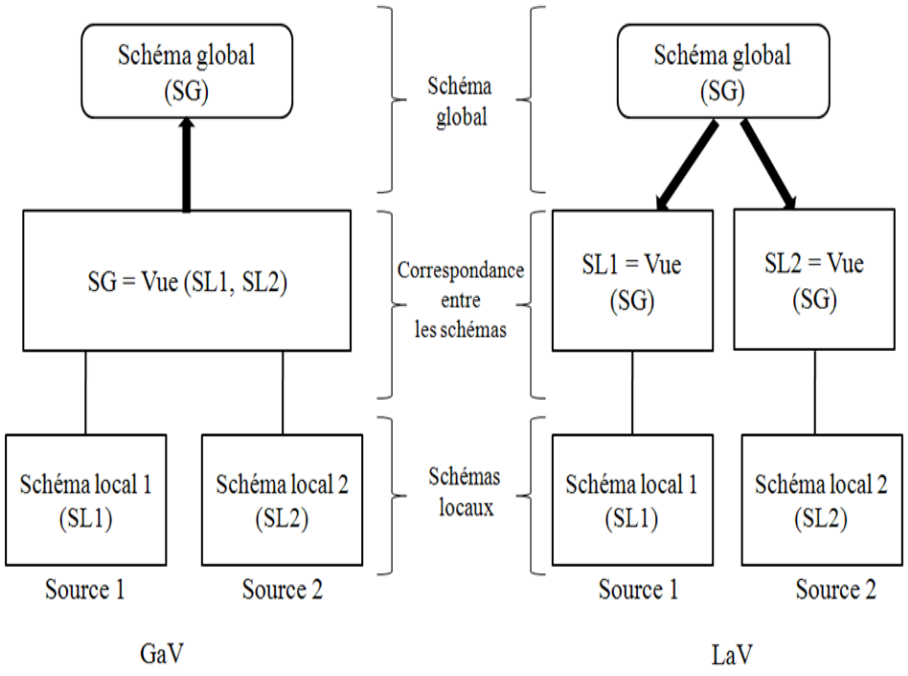

FIGURE 2.5 – Approches GaV et LaV

sont définies comme des vues sur les relations des schémas locaux. Parmi les SID utilisant l'approche GaV, nous citons les systèmes *Tsimmis* [44], *Momis* [20], et *Whips* [81].

2.2.2.2 Approche LaV

L'approche *LaV* est l'approche duale de GaV. Elle définit les schémas des sources à intégrer en fonction du schéma global. Les relations des schémas des sources sont définies comme des vues sur les relations globales. Parmi les SID utilisant l'approche LaV, nous citons les systèmes *InfoMaster* [63] et *Picsel* [73].

Ce deuxième critère est important pour deux problématiques : la *réécriture des requêtes* (réécriture des requêtes utilisateurs sur les sources et construction de la réponse) et la *scalabilité du système d'intégration* (ajout ou suppression de sources de données) [16]. L'approche GaV facilite la réécriture des requêtes mais l'approche LaV assure une meilleure scalabilité du système d'intégration. En effet, une requête utilisateur s'exprime en termes des relations du schéma global, sa réécriture en fonction des schémas des sources dans une approche GaV nécessite un simple dépliement des relations utilisées dans la requête par leurs définitions locales. Cette réécriture, dans une approche LaV, devient un problème complexe nécessitant des inférences [16]. D'autre part, l'ajout d'une nouvelle source de données, pouvant nécessiter une mise à jour du schéma global dans une approche GaV, est facilité dans une approche LaV. Seules les vues des nouvelles sources doivent être rajoutées, sous réserve que le schéma global ait été bien défini initialement.

2.2.3 Le degré d'automatisation du processus d'intégration

Ce dernier critère permet de distinguer les approches d'intégration selon le degré d'automaticité du processus d'intégration. Notons que cette automaticité concerne la résolution ou l'élimination des conflits sémantiques entre les sources durant le processus d'intégration. On distingue les approches *manuelles*, *semi-automatiques* et *automatiques*.

2.2.3.1 Les approches manuelles

Les premières approches proposées étaient des approches *manuelles*. Ces approches permettent d'automatiser l'intégration des données au niveau syntaxique. Les conflits sémantiques sont gérés manuellement et nécessitent la présence d'un expert humain pour interpréter la sémantique des données. Plusieurs SID ont été développés selon cette approche comme les systèmes multi-bases de données [9], la fédération des bases de données [10] ou le système *Tsimmis* [44]. Ces approches manuelles deviennent impraticables lorsque le nombre de sources de données à intégrer est important, ou lorsque les sources évoluent fréquemment.

2.2.3.2 Les approches semi-automatiques

Afin d'apporter davantage d'automatisation dans la résolution des conflits sémantiques, plusieurs travaux se sont tournés vers les ontologies. Une ontologie permet de fournir la sémantique des concepts d'un domaine de manière formelle.

Les approches *semi-automatiques* reposent sur des ontologies *linguistiques* et permettent d'automatiser partiellement la gestion des conflits sémantiques. Les ontologies linguistiques traitent des termes, et non des concepts. Ceci peut créer des conflits de noms. *Momis* [20] est un exemple de projet reposant sur l'ontologie linguistique Wordnet pour l'intégration des sources. Wordnet [11] est une base de données lexicale développée par des linguistes du laboratoire des sciences cognitives de l'université de Princeton.

2.2.3.3 Les approches automatiques

Les approches *automatiques* consistent à associer aux données des sources une ontologie *conceptuelle* qui en définit le sens (figure 2.6). L'utilisation des ontologies conceptuelles a été identifiée par de nombreux travaux comme la clé de l'automatisation du processus d'intégration [16]. La sémantique du domaine est ainsi spécifiée formellement à travers des concepts, leurs propriétés ainsi que les relations entre les concepts. La référence à une ontologie permet d'éliminer automatiquement les conflits sémantiques entre les sources. La connexion entre les sources et l'ontologie peut se faire de plusieurs manières [210] : définition des termes de la BD par les concepts

9. Un système multi bases de données vise à construire un système d'accès à de multiples BD. L'utilisateur spécifie sa requête textuellement en précisant les sources interrogées. Toutes les hétérogénéités sont traitées par l'utilisateur [16].

10. La fédération des bases de données vise à fournir un schéma global représenté par l'union de toutes les BD. Des schémas externes sont définis pour chaque BD, qui seront ensuite convertis en un formalisme commun (relationnel par exemple), et intégrés dans un schéma global. La construction du schéma global est faite manuellement [16].

11. http ://wordnet.princeton.edu/

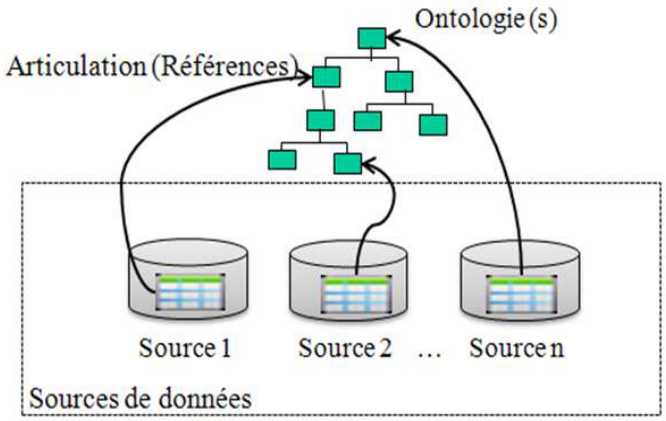

FIGURE 2.6 – Intégration des sources à base d'ontologies

ontologiques (Projet *Buster* [207]), annotation des données (Projet *SHOE* [84]), enrichissement de la source par des règles logiques (Projet *Picsel* [73]), etc. Trois principales structures présentées en figure 2.7 sont utilisées dans un système d'intégration à base ontologique [210] :

- Structure 1 : structure à base d'une ontologie unique,
- Structure 2 : structure à base d'ontologies multiples,
- Structure 3 : structure à base d'une ontologie partagée.

2.2.3.3.1 Structure à base d'une ontologie unique

Dans les SID à base d'une *seule ontologie*, chaque source de données est liée à une unique ontologie (Projets *COIN* [67] et *Picsel* [73]). Ces approches peuvent être utilisées uniquement lorsque les sources à intégrer fournissent globalement la même vue du domaine, autrement il sera difficile de trouver un vocabulaire minimal commun partagé entre les sources. De plus, des changements au niveau des sources peuvent affecter la conceptualisation du domaine représenté par l'ontologie [210]. Les sources de données ne peuvent alors évoluer indépendamment, leur autonomie schématique [12] est donc restreinte. La scalabilité du SID est également limitée par la portée de l'ontologie où l'ajout de nouvelles sources peut nécessiter la redéfinition de l'ontologie globale.

2.2.3.3.2 Structure à base d'ontologies multiples

Une deuxième structure de systèmes d'intégration à base *d'ontologies multiples* a été proposée pour gérer les inconvénients de la première structure. Dans ces systèmes, chaque source de données définit sa sémantique par une ontologie locale. Les ontologies locales sont mises en correspondance deux par deux. Le principal inconvénient de cette approche est la difficulté d'effectuer les mappings. En effet, si le nombre de sources est de N sources, le nombre de mapping est dans ce cas égal à N(N-1)/2.

12. L'autonomie schématique d'une source de données est sa capacité à faire évoluer son schéma local indépendamment.

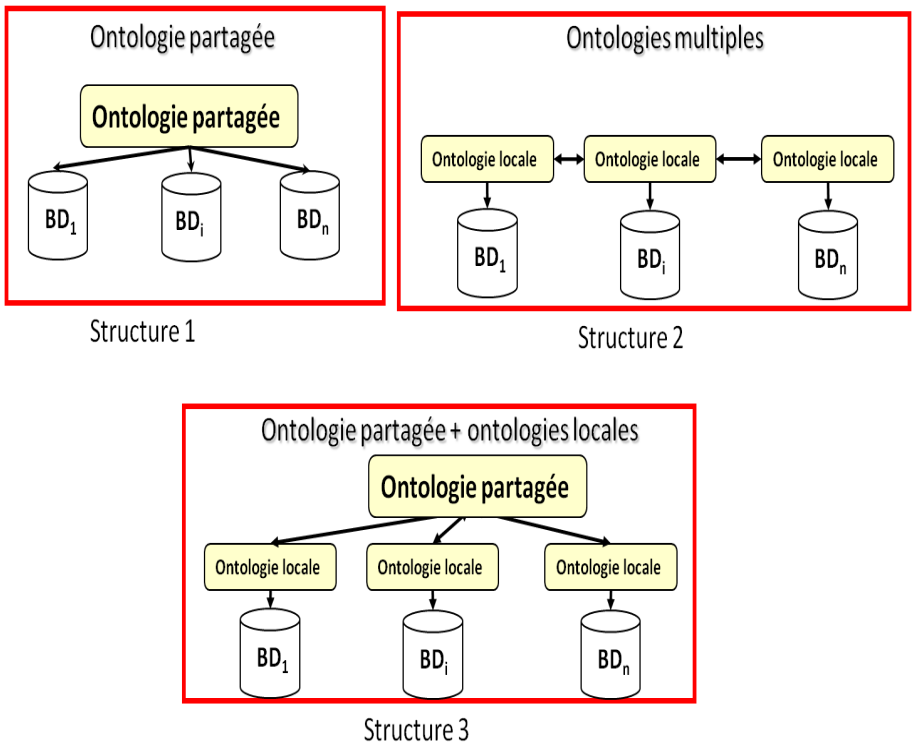

FIGURE 2.7 – Approches d'intégration à base d'ontologies conceptuelles

2.2.3.3.3 Structure à base d'une ontologie partagée

Dans la troisième structure des systèmes d'intégration à base d'une *ontologie partagée*, on retrouve les notions d'ontologie globale et d'ontologies locales. Chaque source est décrite sémantiquement par sa propre ontologie, qu'on appelle ontologie locale. Cette dernière peut être contenue au sein de la BD, c'est le cas des BDBO. Afin de faciliter le mapping entre les ontologies locales, elles sont mises en correspondance avec une ontologie partagée modélisant un domaine particulier, qu'on appelle ontologie globale. L'intégration dans cette structure passe par les étapes suivantes [16] : (1) la définition de l'ontologie locale pour chaque source, (2) la mise en correspondance ou l'alignement entre les ontologies locales des sources et l'ontologie partagée en établissant des relations sémantiques entre leurs concepts. L'alignement entre les ontologies locales et l'ontologie partagée permet d'identifier les relations entre les entités des différentes ontologies.

L'alignement entre les ontologies locales et l'ontologie partagée peut être effectué de deux manières : *à postériori* ou *à priori* [16]. Dans une approche *à postériori*, les correspondances entre les différentes ontologies utilisées sont définies à postériori, éventuellement en utilisant des algorithmes de mapping. Chaque source possède sa propre ontologie locale. Les ontologies locales sont définies de manière indépendante les unes des autres. Une fois ces ontologies locales définies, elles sont alignées avec l'ontologie globale. Une approche *à priori* repose sur une ontologie globale préexistante. Cette approche est utilisée lorsque les administrateurs souhaitent une communication directe entre les sources de données à intégrer, i.e, il existe une volonté de fusion à priori dès la conception de la source de données. Chaque source possède une ontologie locale qui définit sa

sémantique. Ces ontologies locales sont définies à partir de l'ontologie globale en la référençant.

2.2.4 Enrichissement de la classification

Nous enrichissons cette classification par deux critères sur lesquels reposent nos contributions pour ce chapitre : le *niveau d'abstraction* des schémas du SID, et le *type de déploiement* utilisé.

2.2.4.1 Le niveau d'abstraction des schémas du SID

Le *niveau d'abstraction* des différents schémas utilisés peut être défini au niveau logique ou conceptuel. Certains SID utilisent une représentation *logique* (généralement relationnelle) des différents schémas locaux et du schéma global. Les mappings sont dans ce cas définis en utilisant les opérateurs de l'algèbre relationnelle comme la projection, la sélection, les jointures, etc. On retrouve par exemple ce type de mappings dans les systèmes de *BD distribuées*, où un schéma global est décomposé en plusieurs fragments (horizontaux, verticaux ou mixtes). D'autres SID définissent des mappings plus complexes où un schéma local est défini comme une vue sur le schéma global (et vice versa). La définition des mappings entre des schémas logiques doit prendre en compte plusieurs contraintes d'implémentation, et impose un choix de déploiement du schéma final, généralement similaire à la représentation des schémas sources. Cependant, plusieurs SID comme les systèmes d'ED peuvent être déployés sur différentes plateformes logiques (MOLAP, ROLAP, HOLAP) et nécessitent davantage de flexibilité. Pour ce faire, les SID doivent être définis à un niveau d'abstraction conceptuel indépendant de toute contrainte d'implémentation.

Certains SID optent pour une représentation *partiellement conceptuelle*, où seul le schéma global est défini par son modèle conceptuel ou ontologique. Différent travaux comme [37], [38], [39] définissent des SID dont le schéma global est défini respectivement par : un modèle Entités/Associations enrichi et formalisé en langage *DLR*, un modèle conceptuel formalisé en langage *DL-Lite*$_A$ et un modèle conceptuel formalisé en langage *OWL2DL*. Les sources sont définies par leurs schémas relationnels. Dans ces travaux, l'objectif de la définition du schéma global au niveau conceptuel est d'assigner de la sémantique aux schémas des sources et de faciliter ainsi le processus d'intégration. Un deuxième objectif est de faciliter l'interprétation des requêtes des utilisateurs.

D'autres SID définissent des systèmes *complètement conceptuels* où le schéma global et les schémas locaux sont définis au niveau *conceptuel*. Les travaux suivants [35, 79, 16] proposent des SID dont les différents schémas sont définis par des modèles conceptuels formalisés respectivement par les langages DLR, OWL-DL et PLIB. Tous ces langages (DLR, DL-Lite$_A$, OWL2DL, OWL-DL) présentent des fragments du formalisme des LD. Certains de ces langages comme OWL2DL et OWL-DL sont des langages ontologiques. L'objectif des deux travaux [35, 79] est de fournir un Framework d'intégration et d'interopérabilité des ontologies dans un système médiateur. L'objectif de l'étude proposée par [16] est de fournir une méthode complètement automatique d'intégration des sources selon une approche dirigée par les ontologies.

Les SID considérant des BDBO comme sources de données entrent dans cette dernière catégorie de SID dont les schémas des sources sont représentés par leurs modèles conceptuels (l'ontologie locale contenue dans la BD), et le schéma global est représenté par l'ontologie partagée référencée par les BDBO.

2.2.4.2 Le type de déploiement du SID

La majorité des travaux portant sur les SID ne prennent pas en compte le critère de déploiement du SID. Les SID définissant leurs schémas des sources au niveau logique imposent un schéma de déploiement qui suit la représentation des sources. L'introduction d'un schéma global conceptuel apporte une facilité d'intégration des données, mais ne facilite pas le déploiement multiple du schéma du SID. La définition du SID à un niveau complètement conceptuel où l'ensemble des schémas du SID sont décrits pas des formalismes conceptuels (le schéma global, les schémas des sources et les mappings) permet de fournir un déploiement multiple du SID. Cependant, aucun des travaux cités proposant des SID complètement conceptuels ne s'orientent vers cet objectif. Ils ne fournissent donc aucun moyen pour permettre ce type de déploiement *à la carte*.

Le SID que nous proposons est un système d'intégration *matérialisé* de type ED, dont le sens du mapping est de type *GaV*, dans lequel le processus d'intégration se fait de manière *automatique*. Une approche *LaV* est également acceptée avec la possibilité de réécriture de certaines assertions de mapping. Le système d'ED que nous proposons est défini complètement au niveau *conceptuel ontologique*. Nous fournissons une approche permettant le déploiement de l'ED à la carte. Pour ce faire, nous avons défini un Framework d'intégration générique <G,S,M> basé sur le formalisme des LD, que nous détaillons ci-dessous. Nous décrivons ensuite les étapes de conception de l'ED.

2.3 Définition du système d'ED déployé à la carte

Dans ce qui suit, nous présentons en premier lieu l'étude de cas illustrant la méthode proposée. Nous présentons ensuite le Framework générique d'intégration formalisant les sources de données considérées et les mappings avec le schéma global. Nous présentons enfin la méthode de conception proposée.

2.3.1 Etude de cas

Nous utilisons dans notre étude de cas l'ontologie du banc d'essai ontologique LUBM [13] (*Lehigh University Benchmark*), illustrée en figure 2.8, représentant une ontologie du domaine universitaire. Nous considérons cette ontologie comme le schéma global G référencé par un ensemble de sources de données. Nous considérons aussi les buts illustrés en figure 2.9 comme l'ensemble des besoins collectés auprès des utilisateurs.Ces besoins sont inspirés des requêtes fournies au niveau du banc d'essai LUBM que nous avons reformulée sous forme de besoins stratégiques (sous forme de buts).

Le choix du banc d'essai LUBM s'explique d'une part par le fait que ce dernier présente un banc d'essai ontologique, il s'applique ainsi aisément à notre méthode. D'autre part, LUBM fournit un générateur de données permettant de générer des données de différentes tailles. Ceci nous sera utile dans l'étape d'expérimentation où nous testons la faisabilité et la scalabilité de notre méthode d'intégration.

13. http ://swat.cse.lehigh.edu/projects/lubm/

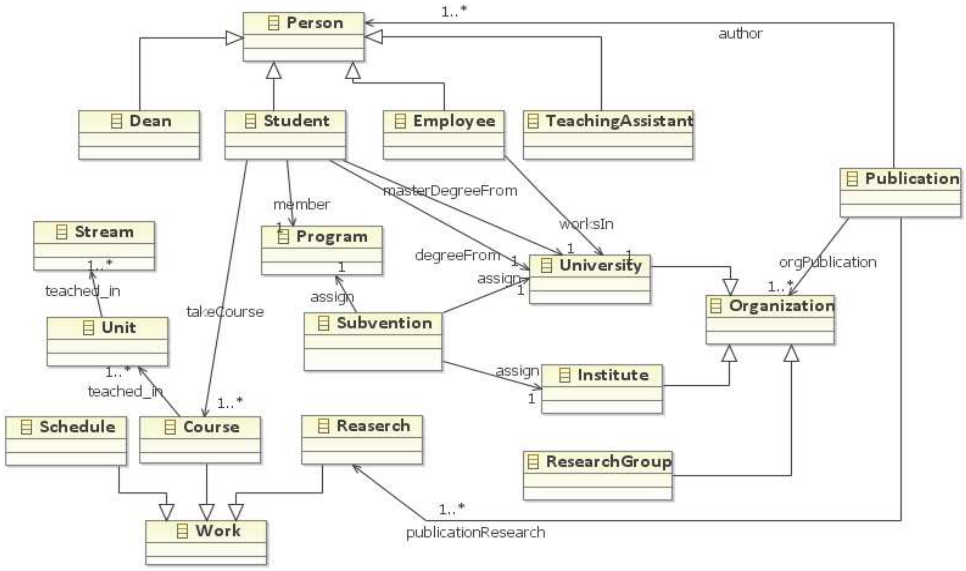

FIGURE 2.8 – Schéma de l'ontologie du banc d'essai LUBM

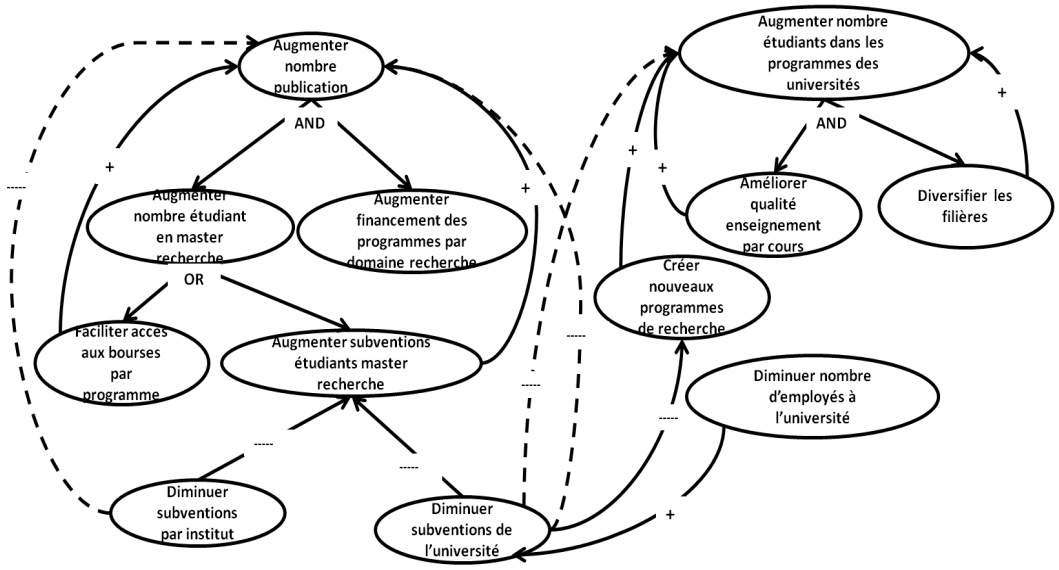

FIGURE 2.9 – Le graphe de buts du domaine universitaire

2.3.2 Formalisation du Framework d'intégration générique

Pour représenter la diversité des structures des sources (conventionnelles et sémantiques) intervenant dans les systèmes d'intégration, nous proposons un Framework d'intégration représentant un SID défini au niveau conceptuel. Nous définissons ce Framework sur la base du formalisme des LD.

Un système d'intégration de données est basé sur un schéma global et un ensemble de sources. Les sources contiennent les données, alors que le schéma global fournit une vision globale de l'ensemble des sources. Un aspect important dans la conception d'un système d'intégration est la spécification des correspondances (mappings) entre les schémas des sources et le schéma global. Le Framework d'intégration est formalisé par le triplet <G,S,M> où G est le schéma global, S représente les sources de données S = S1, . . ., Sn et M un ensemble de mappings. Chaque élément du Framework est formellement défini comme suit.

2.3.2.1 Le schéma global G

Le schéma global est représenté dans notre méthode par l'*ontologie partagée* par les sources. Pour formaliser le schéma global, nous reprenons la formalisation proposée dans le chapitre précédent. Le schéma global G est représenté par son modèle conceptuel ontologique (MO) basé sur le formalisme des LD défini par le 5-uplet : <C, R, Applic(C), Sub(C), Ref(C), Ref'(R), Formalisme>, présenté en section 1.3.3.2.1.

2.3.2.2 Les sources S

Chaque source S_i est définie comme suit : S : <MO_i, I, Sch, Pop, SM_{MO_i}, SM_{Inst}, Ar>, tel que :

- MO_i : représente le modèle conceptuel MO de la source S, formalisé comme décrit ci-dessus : $< C_i, R_i, Applic_i(C), Sub_i(C), Ref_i(C), Ref'_i(R), Formalisme_i >$
- I : représente l'ensemble des instances (les données) des sources.
- $Sch : C_i \rightarrow 2^{R_i}$ est une fonction qui associe à chaque classe de C_i l'ensemble des rôles pour lesquels les instances de cette classe sont évaluées.
- $Pop : C_i \rightarrow 2^I$, est une fonction qui associe à chaque classe ses instances de l'ensemble I.
- Modèle de stockage (MS_{MO_i}) : représente le schéma de stockage du modèle conceptuel (vertical, horizontal, etc.) dans la source. Ce modèle peut être à nul si la source n'est pas définie par son modèle conceptuel.
- Modèle de stockage (MS_{Inst}) : représente le schéma de stockage des instances.
- Modèle d'architecture (Ar) : représente le type d'architecture de la base (Type I, II ou III). Les BD classiques correspondent à une architecture de type I. Les BDBO peuvent être définies selon les trois types d'architecture.

Par exemple :

- La BDBO *Sor* d'IBM est définie comme suit : <MO (OWL-DL), I, Sch et Pop (obtenue par des requêtes SPARQL), relationnel binaire, relationnel horizontal, Type II>

2.3.2.3 Les mappings M

Des assertions de mappings sont définis entre le schéma global et les schémas locaux des sources comme suit : M :< $MapSchemaG$, $MapSchemaS$, $MapElmG$, $MapElmS$, $Interpretation$, $SemanticRelation$, $Strength$, $Type$ >. Cette formalisation est basée sur deux travaux : la formalisation proposée dans [41] et le méta-modèle de mapping proposé dans [167]. Nous n'étudions pas ici la découverte des mappings, nous les supposons vérifiés et validés par le concepteur. Cette question de découverte des mappings a été largement étudiée dans le domaine d'alignement/matching de schémas. Nous décrivons chaque élément des assertions mapping.

- $MapSchemaG$ et $MapSchemaS$: présentent respectivement le schéma source et cible sur lesquels seront définis les mappings. Ces deux schémas sont définis par leur modèle conceptuel (MO).
- $MapElmG$ et $MapElmS$: présentent respectivement l'élément (classe, rôle ou expression) concerné par le mapping des schémas MapSchemaG et MapSchemaS.
- $Interpretation$: définit l'interprétation *intentionnelle* (niveau schéma) ou *extensionnelle* (niveau instances) du mapping.
- $SemanticRelation$: définit la relation sémantique entre les deux éléments $MapElmG$ et $MapElmS$. Trois types de relations sont définies [167] : *Equivalence*, *Appartenance* (exacte ou complète) ou *Chevauchement*. Nous rappelons qu'une relation d'*équivalence* indique que les éléments liés par le mapping présentent la même sémantique dans le domaine décrit. Une relation d'*Appartenance exacte* indique que le premier élément du mapping est plus spécifique que le deuxième élément. Une relation d'*Appartenance complète* indique que le deuxième élément du mapping est plus spécifique que le premier élément. Une relation de *Chevauchement* indique que les deux éléments peuvent partager une sémantique commune.
- $Type$: indique le *sens du mapping* : GaV, LaV or GLaV (qui peut être déduit de MapElmG et MapElmS).

Avant de présenter la méthode de conception que nous proposons, nous commençons par établir les hypothèses posées pour la méthode.

2.3.3 Hypothèses

Dans notre méthode de conception de l'ED, l'ontologie partagée par les sources joue le rôle du schéma global dans le Framework <G,S,M>. Les hypothèses citées pour la première proposition portant sur le modèle de l'ontologie partagée (cf. section 1.3.1), sont également valables pour cette deuxième proposition. Nous complétons ces hypothèses par les hypothèses posées pour les sources.

2.3.3.1 Hypothèse 1

Chaque source de données participant au processus d'intégration contient un ensemble de données. Les données des sources représentent les instances des classes sources (individus) et sont supposées remplir les hypothèses de mono-instanciation et de typage fort des instances :

2.3.3.1.1 Mono-instanciation

Toute instance du domaine est décrite par son appartenance à une et une seule classe, qui est

la classe de base dans la hiérarchie des classes à laquelle elle appartient (elle appartient bien sûr également à ses superclasses).

2.3.3.1.2 Typage fort des instances

Toute instance du domaine ne peut être décrite que par les propriétés applicables à sa classe de base, c'est-à-dire celles dont le domaine subsume cette classe de base.

2.3.3.2 Hypothèse 2

Chaque source de données S_i visant à être intégrée contient explicitement ses données, son schéma et son modèle conceptuel (MO_i). Chaque source définit également l'articulation entre son modèle conceptuel (MO_i) et le modèle conceptuel du schéma global (MO) à travers des assertions de mappings définies par le Framework. La source S_i s'engage sur le sens et sur le fait d'utiliser les définitions du schéma global MO en respectant le principe d'engagement sur une ontologie de référence, énoncé dans [146] . Plus précisément, s'engager sur une ontologie partagée signifie respecter la contrainte suivante (appelée SSCR : *Smallest Subsuming Class Reference*) :

- Toute classe locale doit référencer, par la relation *OntoSub*, la plus petite classe subsumante existante dans la hiérarchie de référence si ce n'est pas la même que celle de sa propre super classe ; elle ne référence aucune classe que s'il s'agit d'un concept complètement nouveau pour le domaine,
  $OntoSub : C \rightarrow 2^{C_i}$ représente les relations de subsomption entre MO et MO_i qui associent à chaque classe $c \in C$ l'ensemble des classes $c_i \in C_i$ qui sont subsumées directement par c :

$$\forall\, c \in C, OntoSub(c) = \{c_i \in C_i \mid (c \textit{ subsume } c_i) \wedge ((\forall\, c'_i \mid c_i \in Sub_i(c'_i)) \Rightarrow c'_i \notin OntoSub(c)) \wedge ((c' \in Sub(c)) \Rightarrow c_i \notin OntoSub(c'))\}.$$

L'objectif de ce principe est de réduire l'hétérogénéité. Ainsi, si deux classes dans deux sources différentes se réfèrent au même concept, elles doivent le référencer exactement dans une ontologie de référence.

2.3.4 Méthode de conception proposée

Nous fournissons une méthode de conception couvrant les cinq phases du cycle de conception des ED : définition des besoins, modélisation conceptuelle, phase ETL, modélisation logique et une dernière phase de modélisation physique. Notre contribution consiste revisiter ce cycle en définissant la phase ETL au niveau *conceptuel ontologique*, dans le but de permettre un déploiement de l'ED à la carte. La méthode que nous proposons complète la méthode présentée dans le chapitre précédent par la phase de conception ETL défini au niveau conceptuel. Nous présentons l'ensemble des étapes en rappelant brièvement les étapes présentées dans le chapitre précédent. L'architecture générale de la solution proposée est présentée dans la figure 2.10.

2.3.4.1 Définition des besoins

Dans notre méthode de conception du schéma de l'ED, l'ontologie partagée par les sources joue le rôle du schéma global dans le Framework <G,S,M>. Les besoins collectés auprès des utilisateurs et formalisés selon le modèle de buts pivot, sont projetés sur le schéma global G. Cette

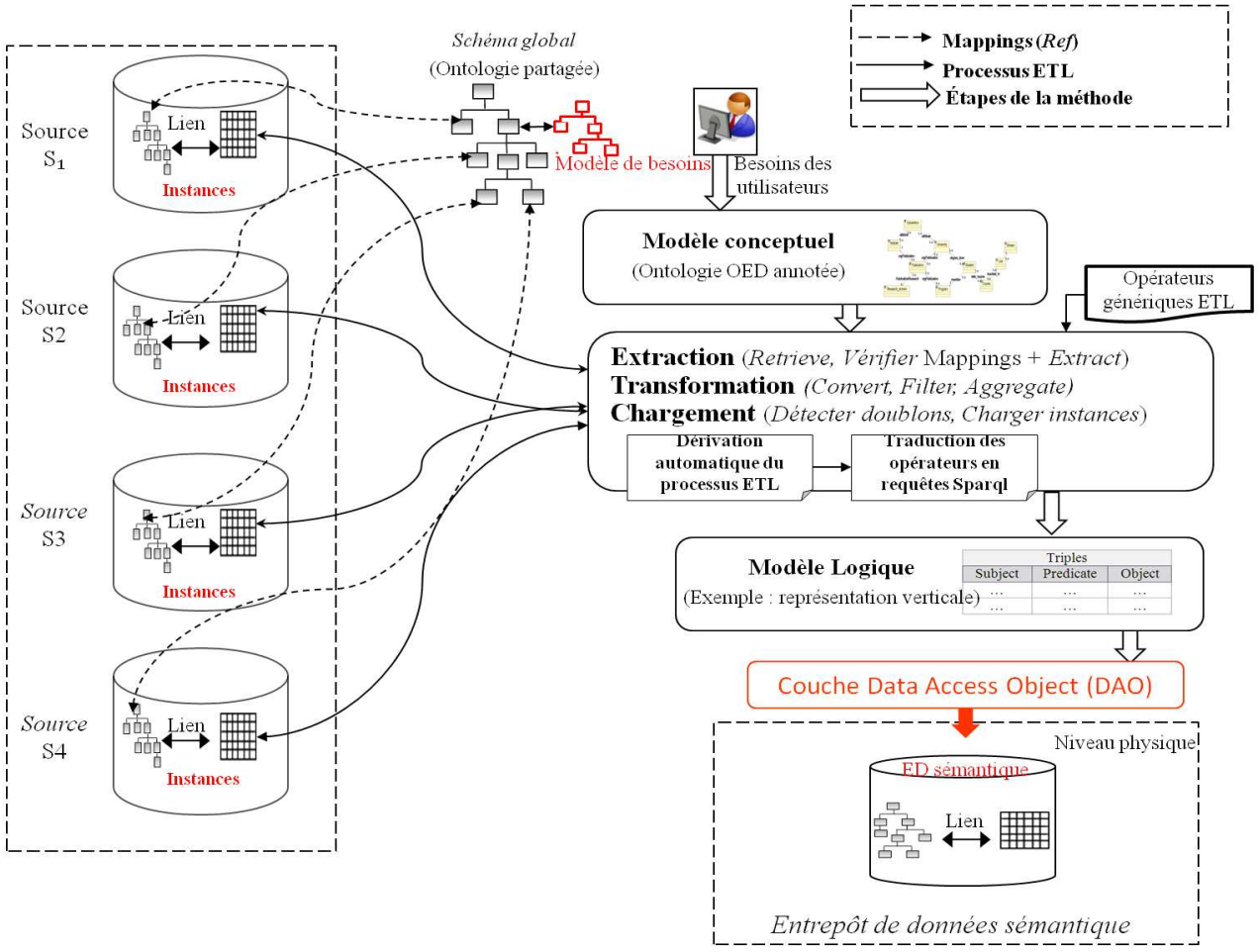

FIGURE 2.10 – Méthode de conception proposée

étape permet de déterminer l'ontologie intégrante, ou l'ontologie de l'ED (OED) qui contient les entités et propriétés pertinentes que devra représenter l'ED final (cf. section 1.3.4.1).

2.3.4.2 La modélisation conceptuelle

L'ontologie OED est extraite comme module représentatif des besoins à partir de l'ontologie partagée, selon les trois scénarios d'extraction suivants :

- $OED = OP$: l'OP correspond exactement aux besoins des utilisateurs.
- $OED \subset OP$: l'OED est extraite à partir de l'ontologie partagée et couvre tous les besoins des utilisateurs.
- $OED \supset OG$: L'ontologie partagée ne couvre pas tous les besoins spécifiques des utilisateurs. L'ontologie locale est extraite, et peut être étendue par de nouvelles classes ou propriétés. Un travail de mapping doit dans ce cas être fait par le concepteur consistant à lier les nouveaux concepts aux concepts sources.

Les mécanismes de raisonnement de l'ontologie OED sont exploités afin de valider le modèle de l'OED et afin de raisonner sur les buts exprimés pour identifier de nouveaux concepts Faits et Dimensions. Cette ontologie sera annotée par les concepts multidimensionnels comme défini dans l'*algorithme* 1. Dans ce qui suit, nous introduisons le processus ETL au niveau conceptuel. La phase ETL permettra d'alimenter cette ontologie OED par les données extraites des sources. Nous appliquons ensuite les mécanismes de raisonnement de l'ontologie afin de vérifier sa consistance (cf. section 1.3.4.2).

2.3.4.3 La phase de conception ETL

La phase ETL correspond au processus d'extraction, transformation et chargement des données sources dans un ED sémantique dont le schéma correspond à l'ontologie locale (OED). Pour effectuer le processus ETL, nous nous sommes basés sur un ensemble d'opérateurs conceptuels définis par [181] (étude référence du domaine ETL), présentant les opérateurs génériques qui sont habituellement rencontrés lors d'un processus ETL. Puisque le schéma de l'ED est représenté par une ontologie locale, nous allons fournir la traduction de ces opérateurs ETL conceptuels en des requêtes ontologiques. Nous présentons dans ce qui suit ces opérateurs et nous les définissons ensuite au niveau ontologique dans le but de les appliquer sur l'ontologie OED.

2.3.4.3.1 Définition des opérateurs ETL

- **EXTRACT(S, C)** : cet opérateur permet d'extraire les données d'une classe selon une condition. Au niveau ontologique, cet opérateur permet la sélection et l'extraction des instances associées à la classe C de la source S. Les instances appropriées sont identifiées selon la contrainte définie au niveau de la classe de l'OED (par exemple, des contraintes de cardinalités, de valeurs, etc).
- **RETRIEVE(S, C)** : cet opérateur permet de rechercher les données pertinentes au niveau des sources. Au niveau ontologique, cet opérateur correspond à la récupération des instances associées à la classe C de la source S.
- **MERGE(S, I)** : cet opérateur permet la fusion des données ayants des attributs compatibles à partir d'une même source. Au niveau ontologique, cet opérateur correspond à la

fusion des instances associées à des classes appartenant à la même source S.

- **UNION (C, C')** : cet opérateur est utilisé pour fusionner deux entités appartenant à différentes sources de données. Au niveau ontologique, ce mécanisme correspond à l'union des instances associées aux classes C et C' appartenant respectivement aux sources S et S'. Ces classes doivent être identiques ou ayant la même super classe.
- **JOIN (C, C')** : cet opérateur est utilisé pour unir deux entités liées par des attributs communs. La jointure est définie au moyen d'une condition restrictive. Au niveau ontologique, la jointure des instances associées aux classes C et C' est réalisée lorsque ces classes sont liées par une propriété. La restriction est définie au niveau du rang de la propriété.
- **STORE(S,C,I)** : cet opérateur permet le chargement des données dans le schéma cible. Au niveau ontologique, ce mécanisme correspond à l'affectation des instances I, récupérées à partir des sources S, à la classe C de l'ontologie OED.
- **DD(I)** : ce mécanisme permet la détection et la suppression des doublons (les instances en double).
- **FILTER(S, C, C')** : cet opérateur permet la sélection des données qui vérifie un certain nombre de contraintes. L'objectif est de charger uniquement les données requises dans l'ED. Au niveau ontologique, l'opérateur Filter permet la sélection des instances associées à la classe C' de la source S, autorisant uniquement les instances correspondant à la contrainte spécifiée par la classe C de l'ontologie OED. Les contraintes sont définies au niveau de l'ontologie OED sous la forme d'axiomes, par exemple par des cardinalités min et max ou par des énumérations.
- **CONVERT(C, C')** : la conversion est le mécanisme utilisé pour modifier les types et formats des données, plus précisément le format des attributs associés aux entités. Au niveau ontologique, la conversion correspond à la modification du format des instances associées à la classe C' d'une source de données vers le format spécifié par la classe C de l'OED. La conversion s'effectue sur la base des propriétés (DataType pour une ontologie OWL).
- **AGGREGATE (F, C, C')** : l'agrégation est le mécanisme utilisé pour regrouper des données à des fins d'analyse. Ce mécanisme agrège les données selon certaines fonctions d'agrégation comme : SUM, AVG, MAX, MIN, COUNT. Au niveau ontologique, l'agrégation consiste à appliquer la fonction F sur les instances associées à la classes C' d'une source, et les associer à la classe C de l'ontologie OED. Le résultat est défini par la classe C.
- **MINCARD(P,min)** : cet opérateur permet d'appliquer une contrainte sur les instances de cardinalité minimale ($\leq$). Au niveau ontologique, la contrainte est appliquée sur la propriété P. Elle est définie par un axiome *min-cardinality*.
- **MAXCARD(P,max)** : cet opérateur permet d'appliquer une contrainte sur les instances de cardinalité maximale ($\geq$). Au niveau ontologique, la contrainte est appliquée sur la propriété P de l'OED. Elle est définie par un axiome *max-cardinality*.

2.3.4.3.2 Proposition de l'algorithme ETL

En se basant sur les opérateurs conceptuels génériques présentés ci-dessus, nous avons proposé l'*algorithme* 4 permettant d'alimenter le schéma de l'ontologie de l'ED (OED) par les données

des sources. Ce processus d'intégration repose sur la sémantique des mappings entre le schéma de l'ontologie partagée (G) et les schémas conceptuels des sources (S_i). Chaque source définit son modèle conceptuel MO_i. Le processus ETL se base sur les mappings définis au niveau *intensionnel* entre les classes du schéma global et les classes des schémas sources. En se référant au Framework proposé <G,S,M>, quatre types de mappings sémantiques sont identifiés que nous rappelons :

- *Mappings d'équivalence*($C_G \equiv C_{Si}$) : un mapping d'équivalence indique que les éléments liés par le mapping présentent la même sémantique dans le domaine décrit. Pour ce type de mapping, les contraintes définies au niveau de la classe C_G du schéma global sont identiques à celles définies sur la classe C_{Si} de la source, aucune transformation des instances n'est effectuée. Les instances sont extraites à partir des sources, fusionnées, unifiées ou jointes, ensuite chargées dans le schéma cible de l'ED.
- *Mappings d'appartenance (exact)* ($C_{Si} \subset C_G$) : une relation d'appartenance exacte indique que le premier élément du mapping est plus spécifique que le deuxième élément. Pour ce type de mapping, les instances de la classe source vérifient toutes les contraintes définies au niveau du schéma global G. Aucune transformation des instances n'est effectuée. Les instances sont extraites à partir des sources, fusionnées, unifiées ou jointes ensuite chargées au niveau du schéma cible de l'ED.
- *Mappings d'appartenance (complet)* ($C_{Si} \supset C_G$) : une relation d'appartenance complète indique que le deuxième élément du mapping est plus spécifique que le premier élément. Pour ce type de mapping, les instances des sources vérifient uniquement un sous-ensemble de contraintes requises par le schéma global G. Certaines instances doivent être transformées (converties, filtrées ou agrégées) puis fusionnées, unifiées ou jointes, ensuite chargées dans le schéma cible de l'ED.
- *Mappings de chevauchement* : une relation de chevauchement indique que les deux éléments peuvent partager une sémantique commune. Pour ce type de mapping, nous procédons à l'identification des contraintes requises par les classes du schéma global et qui ne sont pas appliquées au niveau des classes sources. Ce cas peut être considéré comme identique au cas précédent.

En se basant sur les quatre cas de mappings et les règles de transformations présentées, nous avons proposé un algorithme permettant l'extraction, la transformation et le chargement des données dans le schéma cible de l'ED, représenté par l'ontologie OED. Nous expliquons le déroulement de l'algorithme.

La ***première étape*** consiste à identifier, pour chaque classe C_{ED} de l'OED, les classes C_i des sources liées à C_{ED} par des assertions de mappings. Les classes C_i sont celles à partir desquelles les instances doivent être extraites pour peupler la classe C_{ED}.

La ***deuxième étape*** consiste à identifier le type de mapping associant les deux classes C_{ED} et C_i : mapping d'équivalence, d'appartenance ou de chevauchement. Un traitement est associé à chaque type de mapping.

- Dans le cas d'un mapping d'équivalence ou d'appartenance (exact), les instances de la classe source vérifient toutes les contraintes de la classe C_{ED}. Dans ce cas, le processus d'identification des opérations ensemblistes (fusion, jointure, union) est lancé. L'opération ensembliste MERGE est appliquée lorsque les instances associées à différentes classes C_i de la même source sont identifiées et sont fusionnées dans le classe C_{ED}. L'opérateur UNION

est appliqué lorsque des instances associées à des classes C_i de différentes sources sont identifiées et sont unifiées dans le classe C_{ED}. Une condition doit être vérifiée pour effectuer cette union : les classes unifiées doivent avoir une superclasse en commun. L'opérateur JOIN est appliqué lorsque les classes C_i identifiées dans une même source de données sont liées par une propriété (rôle).

- Dans le cas d'un mapping d'appartenance (complet) ou de chevauchement, les instances de la classe source vérifient uniquement un sous ensemble des contraintes requises par la classe C_{ED}. Dans ce cas, le processus d'identification des opérations de transformation (Conversion, filtrage, agrégation) est lancé. Une fois les données transformées, un processus d'identification des opérations ensemblistes est lancé. Des classes temporaires sont créées pour gérer ces transformations. Chaque type de contrainte est géré par la création d'une sous classe (C_{conv}, C_{filt}, C_{agg}) de classe C_{ED} faisant l'objet de la transformation. Ces classes temporaires correspondent à la zone de stockage temporaire (Data Staging Area) requise lors d'un processus ETL [202]. Une fois les transformations appliquées, les classes temporaires sont supprimées. Nous détaillons le processus d'identification des opérations de transformation :
  - L'opérateur d'agrégation AGREGATE est généralement appliqué sur les propriétés mesures des classes associées à une classe C_{ED} annotée comme Fait. La fonction d'agrégation est annotée dans la classe C_{ED} ; elle est récupérée du modèle de but étendant le modèle de l'ontologie.
  - L'opérateur CONVERT est appliqué en identifiant les rôles r_{ED} de la classe C_{ED} qui font l'objet de la conversion. Les rôles des sources qui correspondent au rôle r_{ED} et qui ont comme domaine les classes associées par un mapping à la classe C_{ED} sont identifiées. Les valeurs de ces rôles identifiés sont converties selon le format du rôle r_{ED} (spécifié dans le rang du rôle r_{ED}).
  - L'opérateur FILTER est appliqué lorsque des instances des classes sources ne satisfont pas les contraintes imposées sur les classes de l'ED. Dans ce cas, seul un sous ensemble des instances sera chargé dans la classe C_{ED}. Ces instances sont filtrées dans les deux cas suivants :

    (1) lorsqu'une contrainte de sélection est identifiée, par exemple la classe C_{ED} stocke uniquement les instances des étudiants d'un département Y.

    (2) Lorsqu'une contrainte de cardinalité est identifiée. Dans ce cas, les rôles associés à la classe C_{ED} qui font l'objet du filtrage sont identifiés. La contrainte de cardinalité est appliquée sur les instances de la classe C_{ED}. Par exemple : $C_{ED\_Student} \geq hasBook$ min3.

    L'algorithme nécessite dans ce cas que les assertions de mapping soient identifiées selon une approche GaV. Dans le cas où les mappings sont définis selon une approche LaV, le déroulement de l'algorithme nécessite la réécriture des mappings de telle sorte que les classes de l'OED soient définies en fonction des classes des sources.

La ***troisième étape*** consiste à récupérer les instances I_i associées à la classe source C_i via l'opérateur Retrieve. Les transformations identifiées y sont appliquées.

La ***dernière étape*** consiste à l'unification de l'ensemble des instances des classes C_i liées par des assertions de mapping à la classe C_{ED}, et à leur chargement dans la classe C_{ED}. Les

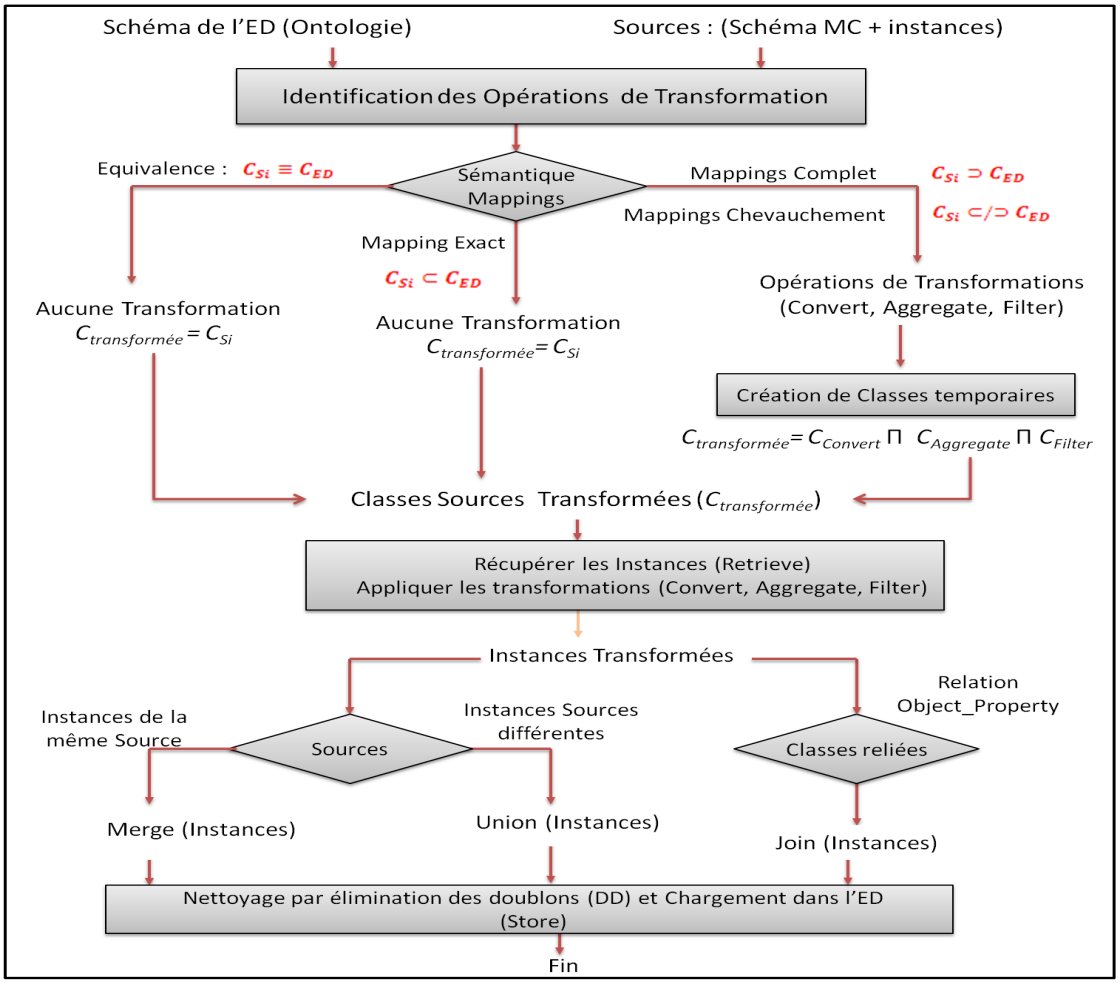

FIGURE 2.11 – Les étapes de l'algorithme ETL

instances doublons sont identifiées et éliminées.

Les étapes citées de l'algorithme sont illustrées dans la figure 2.11, et formalisées dans l'algorithme 4.

L'algorithme proposé effectuant le processus ETL est automatique. Le seul effort fourni par le concepteur consiste à traduire les opérateurs ETL génériques selon le schéma de l'ED cible. Ces opérateurs ETL génériques peuvent être traduits au niveau ontologique en utilisant un langage de requêtes, ou bien au niveau logique selon la représentation choisie de l'ED cible (verticale, binaire, horizontale, etc). Chaque opérateur correspondra à une requête. Dans la section suivante, nous proposons une traduction des opérateurs ETL en utilisant le langage ontologique SPARQL qui est le langage standard recommandé par le consortium W3C.

2.3.4.3.3 Traduction des opérateurs ETL

Dans ce qui suit, nous présentons la traduction des opérateurs ETL génériques au niveau ontologique en utilisant le langage SPARQL. Nous définirons certaines requêtes en utilisant le langage SPARUL ou SPARQL/Update, qui est l'extension du langage SPARQL avec les opérations

```
begin
    Entrée : OED : Ontologie de l'ED (Schéma) et ensemble des sources locales Si
    Sortie : Ontologie intégrante OED (Schéma + Instances)
    pour Chaque classe C de l'ontologie OED faire
        I_OED = φ ; Classe temporaire C' ;
        pour Chaque source Si faire
            si C existe dans Si, elle est nommée Cs alors
                si Cs ≡ C /*Mappings d'équivalence, les instances de Cs vérifient toutes les
                contraintes imposées par C*/ alors
                    C'= IdentiferClasse (Si, C) /*identifier la classe à partir de la source Si*/
                fin
                sinon
                    Cs ⊂ C /* les instances associées à Cs vérifient les contraintes imposées
                    par l'OED et d'autres contraintes */
                    C'= IdentiferClasse (Si, C) /*identifier la classe de la source Si*/
                fin
                sinon
                    Cs⊃ C ou mappings de chevauchement /*les instances de Cs vérifient
                    uniquement un sous-ensemble de contraintes de l'OED*/
                    si format(C) ≠ format(C') alors
                        C'= CONVERT (C, C') /*convertir le format des instances à partir du
                        format de la classe source vers celui de l'OED*/
                    fin
                    si C contient une contrainte d'aggregation définie par une fonction F alors
                        C'= AGGREGATE (F, C, C') /*Agréger les instances en utilisant la
                        fonction F*/
                    fin
                    sinon
                        si C présente une contrainte filtrant certaines instances alors
                            C'= FILTER (Si, C, C') /*Filtrer les instances de C' selon la
                            contrainte C de l'OED*/
                        fin
                    fin
                fin
            fin
        fin
        /*Récupération des instances des classes C' identifiées*/
        pour Chaque classe C' identifiée faire
            I_Si= RETRIEVE (Si, C') /*Récupérer les instances de C'*/
            si Plus d'une instance est identifiée dans la même source alors
                I_OED= MERGE (Si, I_Si) /*Fusion des instances */
            fin
        fin
        si classes C1 et C2 identifiées des sources S et S' ayant la même super classe alors
            I_OED= UNION (C1, C2) /*Union des instances associées aux classes de différentes
            sources*/
        fin
        sinon
            si classes C1 et C2 identifiées des sources S et S' sont liées par la même propriété alors
                I_OED= JOIN (C1, C2) /*Jointure des instances*/
            fin
        fin
        STORE (OED,C, DD(I_OED)) /*Détecter les doublons puis chargement*/
    fin
end
```

Algorithme 4: Algorithme de peuplement du schéma cible de l'ED

d'ajout, de modification et de suppression. Nous supposons que l'espace de nom de l'ontologie de OED est NameSpace_{ED}. Nous illustrons nos propos par des exemples tirés de l'étude de cas présentée (le schéma du banc d'essai LUBM), dont l'espace de nom est le suivant : PREFIX univ-bench1 : <http ://www.lehigh.edu/∼zhp2/2004/0401/univ-bench.owl>

- **EXTRACT** : extraire les instances (Instances) de la classe (Class) de l'OED avec la propriété (DataProperty) selon la condition valeur_condition
  Select ?Instances#
  Where { ?Instance# rdf :type NameSpace_{ED} :Class. ?Instance NameSpace_{ED} :DataProperty valeur_condition}
  Exemple 1 *: extraire les étudiants ayant 25 ans.*
  Select ?student Where { ?student rdf :type univ-bench :Student . ?student univ-bench :age≤25}
- **RETRIEVE** : rechercher les instances (Instances) de la classe (Class) de l'OED
  Select ?Instances# Where { ?Instances# rdf :type NameSpace_{ED} :Class}
  Exemple 2 *: rechercher les instances de la classe Student.*
  Select ?InstanceStudent Where { ?InstanceStudent rdf :type univ-bench :Student}
- **MERGE** : fusionner les instances (Instances) des classes Class1 et Class2.
  Select ?Instances Where {{ ?Instances rdf :type NameSpace_{ED} :Class1}
  Union { ?instance rdf :type NameSpace_{ED} :Class2}}
  Exemple 3 *: fusionner les instances des classes Employee et Student appartenant à la même source.*
  Select ?instance Where {{ ?instance rdf :type univ-bench :Student}
  Union { ?instance rdf :type univ-bench :Employee}}
- **UNION** : unifier les instances (?instance) des classes Class1 et Class2, ayant une super classe en commun.
  Select ?instance
  Where { ?instance rdf :type NameSpace_{ED} :Class1} . {C rdfs :subClassOf NameSpace_{ED} :Class1}
  Union { ?instance rdf :type NameSpace_{ED} :Class2} . {C rdfs :subClassOf NameSpace_{ED} :Class2}

  Exemple 4 *: union des instances des classes Student et Employee ayant la super classe Person en commun.*
  PREFIX univ-bench2 : <http ://www.lehigh.edu/∼zhp2/2004/0401/univ-bench.owl2#>
  Select ?instance
  Where { ?instance rdf :type univ-bench :Student}.{Student rdfs :subClassOf univ-bench :Person}
  Union { ?instance rdf :type univ-bench :Employee}.{Employee rdfs :subClassOf univ-bench :Person}

- **JOIN** : jointure des instances liées par une propriété.
  Select ?instance1 ?instance2
  Where { ?instance1 rdf :type NameSpace_{ED} :Class1.
  ?instance2 rdf :type NameSpace_{ED} :Class2.
  ?instance1 NameSpace_{ED} :P ? instance2}
  Exemple 5 *: jointure des classes Student et Course liées par la propriété takesCourse.*

Select ?instanceStudent ? instanceCourse
Where { ?instanceStudent rdf :type univ-bench :Student.
?instanceCourse rdf :type univ-bench :Course.
?instanceStudent univ-bench :takesCourse ?instanceCourse}

- **DD** : Détecter les doublons et les supprimer.
  Select Distinct ?instance
  Where {{ ?instance rdf :type NameSpace$_{ED}$:Class1} Union { ?instance rdf :type NameSpace$_{ED}$:Class2}}
  Exemple 6 *: détecter et supprimer les instances en double associées aux classes Student et Person.*
  Select Distinct ?instance
  Where { ?instance rdf :type univ-bench :Student} Union { ?instance rdf :type univ-bench :Person}
- **FILTER** : filtrer les instances autorisant uniquement celles vérifiant la propriété P.
  Select ?instance ?P
  where { ?Instance rdf :type NameSpace$_{ED}$:Class . NameSpace$_{ED}$:P ?P .
  FILTER (?P $\geq$ value condition)}
  Exemple 7 *: filtrer les instances de la classe Student autorisant uniquement ceux ayant la propriété âge $\geq$ 16 ans.*
  Select ?instanceStudent ?age
  where { ?instanceStudent rdf :type univ-bench :Student ; univ-bench :age ?age .
  FILTER (?age $\geq$ 16) }
- **AGGREGATE** : agrégation des instances en appliquant la fonction F (F= COUNT, SUM, AVG, MAX).
  Select (Count(?Instance) AS ?count)
  Where { ?Instance rdf :type NameSpace$_{ED}$:Class}
  Group By ?Instance
  Exemple 8 *: calculer le nombre d'étudiants.*
  Select (count(?Student) AS ?count)
  Where { ?Student rdf :type univ-bench :Student}
  Group By ?Student
- **STORE** : chargement des instances associées à une classe de l'ED en utilisant le langage SPARUL.
  INSERT {NameSpace$_{ED}$:Class rdf :ID ?Instance}
  Exemple 9 *: insérer les instances transformées (?Instance) dans la classe Student de l'ED.*
  INSERT {univ-bench :Student rdf :ID ?Instance}

2.3.4.4 La modélisation logique

Le but de notre proposition est de permettre un déploiement logique *à la carte* de l'ED. En effectuant la phase ETL au niveau conceptuel ontologique, nous avons pu définir le schéma conceptuel ontologique de l'ED alimenté par les données des sources. La traduction de ce schéma en un schéma logique peut ainsi se faire selon différentes représentations. Par exemple, le schéma logique relationnel de l'ED est généré à partir de l'ontologie OED intégrante, après la traduction

de ses constructeurs ontologiques en constructeurs relationnels. Nous appliquons les algorithmes proposés dans le chapitre précédent permettant de traduire une ontologie selon les trois principales approches citées : verticale, binaire et horizontale (cf. section 1.3.4.3).

2.3.4.5 La modélisation physique

Une fois l'ED défini (son modèle et ses données), nous proposons d'utiliser une structure de BDBO pour représenter un ED sémantique. Le modèle de l'ED a été défini indépendamment de toute contrainte d'implémentation, l'ED peut donc être déployé selon les trois architectures de BDBO mentionnées (type I, type II, ou type III). Toutes ces architectures permettent de représenter les données de l'ED, et également l'ontologie explicitant la sémantique de ces données. Les modèles définis (modèle ontologique OED et le modèle logique relationnel) seront donc stockés dans l'entrepôt final. Nous rappelons que le méta-modèle de l'ontologie a été étendu par le modèle des besoins, qui sera donc également stocké dans l'ED pour permettre la persistance des besoins. Le choix du modèle de stockage s'applique aussi bien pour le modèle d'ontologie que pour les instances ontologiques (pour peu que le SGBD supporte les modèles de stockage choisis par le concepteur).

D'un point de vue implémentation, nous proposons une solution de déploiement multiple basée sur l'architecture Modèle-Vue-Contrôleur (MVC) [113]. L'algorithme ETL est défini au niveau de la couche contrôleur. Chaque opérateur ETL est implémenté par un objet DAO (Data Access Object). Cette solution permet d'encapsuler tous les accès aux données dans une couche appelée DAO. L'accès aux données se fait via l'interface java DAO. Cette interface reste constante même si la source de données à implémenter change. L'objet DAO joue le rôle d'un adaptateur entre le composant métier et les sources de données à implémenter. Cette solution a été implémentée dans l'outil supportant notre méthode de conception, qui sera présenté à la section suivante.

2.4 Validation : expérimentation et prototypage

Nous présentons dans cette section une instanciation du Framework d'intégration proposé pour des sources BDBO réelles implémentées en utilisant le SGBD sémantique d'*Oracle*. Nous appliquons ensuite les cinq étapes de conception énoncées dans la méthode proposée pour définir un système d'ED à partir de ces BDBO. Nous présentons comme deuxième étape de validation l'outil implémenté pour supporter la méthode proposée.

2.4.1 Instanciation du Framework pour des sources BDBO

La deuxième contribution de la proposition présentée dans ce chapitre est de fournir une méthode de conception complète pour l'intégration des BDBO dans un système d'ED. Nous montrons dans cette section que la méthode définie ci-dessus peut être appliquée à des BDBO par instanciation du Framework <G,S,M> comme suit : <Schéma du banc d'essai LUBM, BDBO Oracle, Mappings à définir>. Le schéma du banc d'essai LUBM est présenté dans la figure 2.8. Nous considérons un ensemble de BDBO Oracle référençant l'ontologie LUBM. Les besoins des utilisateurs sont illustrés dans le graphe de buts de la figure 2.9.

2.4.1.1 Présentation de la DBBO sémantique d'Oracle

Le SGBD *Oracle* a incorporé des supports pour les langages *RDF* et *OWL* dans son système pour permettre à ses clients de bénéficier d'une plateforme de gestion des données sémantiques. Oracle définit deux sous-classes d'OWL DL qu'il exploite dans les processus d'inférence : la classe *OWLSIF* et une classe plus riche *OWLPrime*. Nous utilisons la classe *OWLPrime* qui restreint les constructeurs et axiomes de la LD pour que tous les raisonnements supportés soit décidables. OWLPrime offre les constructeurs suivants : rdfs :domain, rdfs :range, rdfs :subClassOf, rdfs :subPropertyOf, owl :equivalentClass, owl :equivalentProperty, owl :sameAs, owl :inverseOf, owl : TransitiveProperty, owl :SymmetricProperty, owl :FunctionalProperty, owl : InverseFunctionalProperty.

2.4.1.2 Scénario d'expérimentation

Nous considérons l'exemple d'un scénario où un organisme de tutelle impose à différentes universités l'utilisation d'un même vocabulaire. Chaque université procède à la création de sa BDBO Oracle. Le scénario utilisé considère la création de six BDBO Oracle peuplées localement comme illustré dans la figure 2.12. Chaque source extrait un fragment (ou une vue) de l'ontologie LUBM selon des assertions de mappings, puis procède à son peuplement localement, en utilisant les instances du banc d'essai LUBM.

Périodiquement, l'organisme de tutelle doit réaliser des études et effectuer des analyses dans un processus de prise de décision. Les BDBO doivent donc être intégrées dans un ED. Le schéma de l'ED est défini après l'analyse et la spécification des besoins des décideurs et utilisateurs (sous forme de buts). Ces derniers seront sauvegardés au niveau de l'ED.

Afin d'effectuer nos expérimentations et pour mener à bien notre processus de déploiement, nous avons fait varier le nombre d'instances des classes ontologiques. Nous avons pour ce faire utilisé l'outil de génération automatique de données (UBA) fourni par le banc d'essai LUBM. Ces données sont répétables et personnalisables, ce qui nous permet de préciser la génération de nombres aléatoires des données. Les données générées respectent les contraintes de l'ontologie (par exemple, la hiérarchie des classes et les relations entre les instances générées).

L'outil UBA commence par la création des instances de la principale classe Université de l'ontologie. Chaque université se compose de 15 à 25 départements. Dans chaque département (*Department*), nous trouvons différentes catégories de professeurs (*Professor*), d'étudiants (*Student*) dont les étudiants en graduation (*GraduateStudents*), de cours (*Course*), pour ne citer que les principales classes. Nous avons généré six ensembles d'ontologies avec respectivement 1, 3, 6, 9, 12 et 15 universités. Le nombre d'instances dans chaque ensemble ontologique est présenté dans le tableau 2.1.

2.4.1.3 Instanciation du Framework d'intégration

Nous commençons par instancier le Framework d'intégration <G,S,M> proposé, par des BDBO Oracle, créées en référençant le schéma global LUBM selon une approche LaV (le schéma de la source est défini en fonction du schéma global), par différents types de mappings. La création des BDBO sources se fait selon plusieurs scénarios, où chaque BDBO peut être un

Classes ontologiques	Instances ontologiques
1 université	70193
3 universités	210579
6 universités	421158
9 universités	631737
12 universités	842316
15 universités	1052895

TABLE 2.1 – Génération des instances ontologiques du banc d'essai LUBM

simple fragment du schéma global de l'ontologie LUBM, ou une vue sur le schéma global de l'ontologie LUBM.

Le ***schéma global*** est formalisé par son modèle conceptuel ontologique (MO) comme suit : MO_Oracle : <Classes C, Properties P (Datatype Property et Object Property), *Applic(C)* et *Sub(C)* définies par des requêtes Sparql, *Ref(C)* :(Opérateur, *Expression*), *Ref'(R)* :(Opérateur, *Expression*), OWLPrime>.

Ref (resp.Ref') est la fonction qui associe à chaque classe (resp. rôle) une expression définie en utilisant les opérateurs d'équivalence et/ou d'inclusion d'OWLPrime (rdfs :subClassOf, owl : equivalentClass, rdfs :subPropertyOf, owl :equivalentProperty). *Expression* est une expression sur les classes et propriétés de l'ontologie partagée utilisant les constructeurs d'OWLPrime.

Chaque ***source locale*** $\boldsymbol{S_i}$ est instanciée comme suit : S_i : < MO_Oracle, Individuals (triplets), Pop représentée par les tables RDF_link et RDF_values, Vertical, Vertical, type I>. Le modèle de stockage vertical est le schéma relationnel composé d'une table de triplets (sujet, prédicat, object). Par exemple : (Student, type, Class) pour le stockage du schéma, et (Student#1, type, Student), (Student#1, takeCourse, Course#1) pour le stockage des instances.

Les ***mappings*** entre le schéma global et les schémas locaux sont donc définis comme suit : Mapping M :< MO_Oracle de chaque source, MO_Oracle du schéma global, expression sur G (fragment ou vue), Classe de la source S, interprétation intentionnelle, (Equivalent, Containment ou Overlap (owl :SubClassOf et owl :equivalentClass d'OWLPrime)), LaV>.

Les six BDBO sont créées comme suit : deux BDBO sont créées comme des fragments horizontaux, la troisième BDBO est créée comme un fragment vertical sur l'ontologie globale et deux autres BDBO sont créées comme des fragments mixtes. Une classe de l'ontologie locale de la source référence dans ce cas une seule classe de l'ontologie globale. La dernière BDBO est créée comme une vue sur l'ontologie globale où un concept de l'ontologie globale correspond à une expression (utilisant les constructeurs d'OWLPrime) sur les classes et propriétés de l'ontologie globale. Notons que la vérification de la consistance de l'ontologie locale résultante est laissée au soin du concepteur. Nous donnons les définitions suivantes de fragments et de vues ontologiques.

Définition 3 *Un fragment vertical sur l'ontologie globale G est défini comme une projection sur les classes de l'ontologie. Chaque classe hérite de tous ses rôles. La figure 2.13 illustre une fragmentation verticale ontologique.*
En se référant au Framework d'intégration proposé, une fragmentation verticale définie sur l'on-

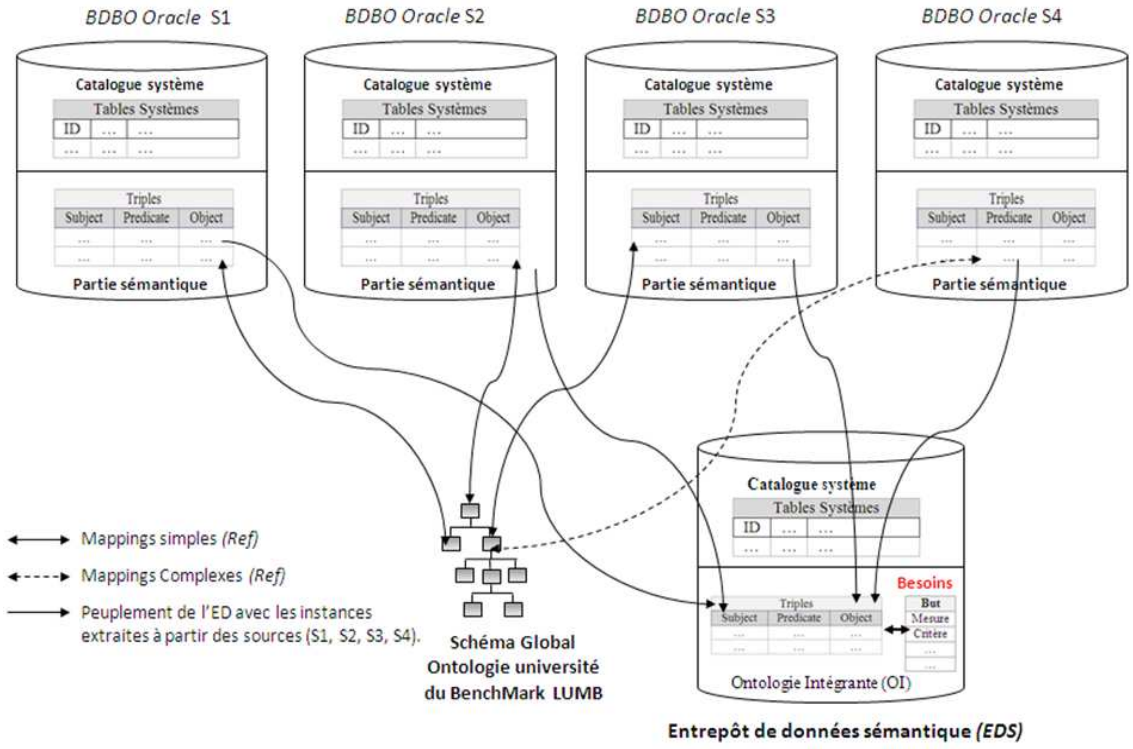

FIGURE 2.12 – Scénario de validation de l'approche proposée avec des BDBO Oracle

tologie globale O :<C : ensemble de classes, R : ensemble de rôles, Applic(C), Sub(C), Ref(C), Ref'(R), Formalisme>, est formellement définie comme suit : $\Pi_{(C_i,C_j,...,C_m)}$ *(C).*

Le premier type de BDBO est obtenu par une projection sur les classes *Person*, *Student*, *GraduateStudent*, *Publication* de l'ontologie LUBM :
$\Pi_{(Person,Student,GraduateStudent,Publication)}$ (C)

Définition 4 *Un fragment horizontal sur l'ontologie globale G est défini comme une restriction sur un ensemble de rôles pour chaque classe de l'ontologie. La figure 2.13 illustre une fragmentation horizontale ontologique. Une fragmentation horizontale sur une ontologie est formellement définie comme suit :* $\sigma_{rest^{(r_i)},rest^{(r_j)},...,rest^{(r_m)}}$ *(C) où* $rest^{(r_i)}$ *représente une restriction sur les rôles* r_i *de la classe C.*

Par exemple pour le deuxième type BDBO, des restrictions sont définies sur les propriétés (*Age* et *EmailAdress*) de la classe *Person* de l'ontologie LUBM :
$\sigma_{Age,EmailAdress}$ $(Person)$

Définition 5 *Un fragment mixte sur une ontologie est défini comme un processus d'application simultanée d'une fragmentation horizontale et verticale sur l'ontologie globale G.*

Par exemple, le troisième type BDBO est un fragment mixte de l'ontologie LUBM, défini par une fragmentation verticale obtenant les classes : *Person*, *Student*, *GraduateStudent* et *Publication*. Une restriction est définie sur deux propriétés de la classe Person : *Age* et *EmailAdress*.

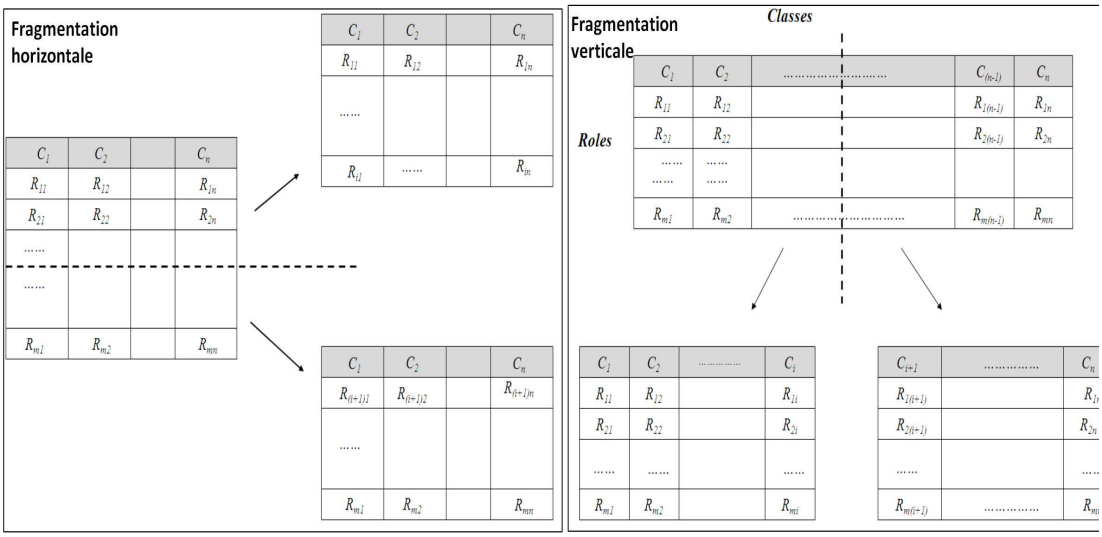

FIGURE 2.13 – Fragmentation verticale et horizontale de l'ontologie globale

Définition 6 *Une vue sur une ontologie est définie par des expressions utilisant les constructeurs de la logique de description OWLPrime. Formellement, une ontologie locale construite par des mappings complexes est définie par : OL : <C', R', Applic'(C'), Sub'(C'), Ref'(C'), Ref'(R'), Ref_{ext}(C'), Formalisme>, où Ref_{ext}(C') représente les références externes entre l'ontologie locale de la source et l'ontologie globale.*

Par exemple, pour le quatrième type de BDBO les trois classes : Person, Student et Publication sont définies comme des vues sur l'ontologie LUBM, comme suit :
S_4 : $Ref_{ext}(Person) = (Student \cup Employee) \cap \forall\ member\ (Person, Organization)$.
S_4 : $Ref_{ext}(Student) = Student \cap \forall takesCourse\ (Person, Course)$
S_4 : $Ref_{ext}(Publication) = Publication \cap \forall PublicationAuthor\ (Publication, GraduateStudent)$

Une fois que le schéma ontologique de chaque source est défini selon ces types de mappings, nous implémentons les BDBO Oracle sources correspondantes à ces schémas. Nous avons utilisé la version d'Oracle 11g version 2. Oracle 11g permet le chargement des données sous forme d'un fichier N-Triple (.nt) selon une représentation verticale. Pour ce faire, nous avons utilisé l'API Jena qui fournit un convertisseur appelé rdfcat. Il permet la transformation des fichiers OWL au format N-Triple. Le nombre d'instances obtenu pour chaque ensemble ontologique est représenté dans le tableau 2.2 complétant le tableau 2.1 par la colonne *Instances N-triples*. Pour chaque cas (ligne) du tableau, ces instances N-triples sont chargées dans les six BDBO en utilisant l'utilitaire *Oracle SQL Loader*. Pour nos expérimentations, nous testons dans ce qui suit notre méthode pour chacun des cas du tableau.

2.4.2 Application de la méthode de conception sur le Framework défini

Nous déroulons la méthode proposée (les cinq étapes) en prenant comme entrées les éléments du Framework instancié par les BDBO.

Classes ontologiques	Instances ontologiques	Instances N-Triple
1 université	70193	84231
3 universités	210579	259012
6 universités	421158	505389
9 universités	631737	770719
12 universités	842316	1027625
15 universités	1052895	1295060

TABLE 2.2 – Génération des instances ontologiques du banc d'essai LUBM

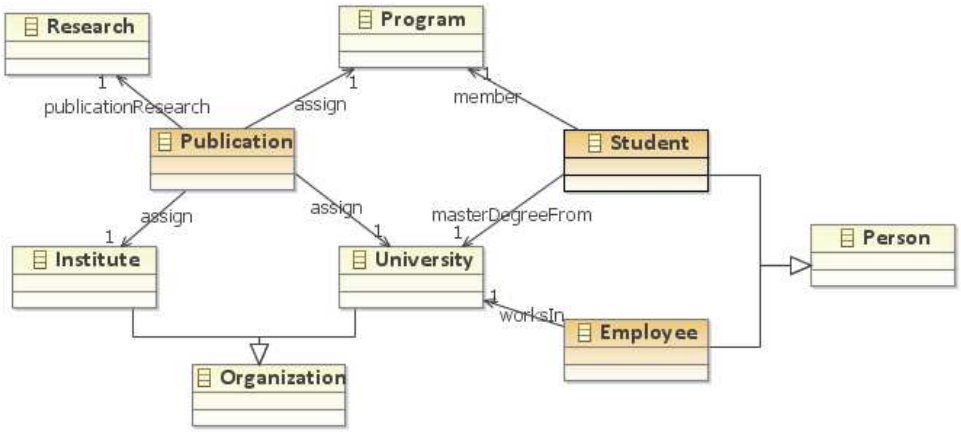

FIGURE 2.14 – Le modèle multidimensionnel obtenu

2.4.2.1 Définition des besoins et du modèle conceptuel

Les besoins considérés dans cette étude sont les buts représentés dans la figure 2.9. Ces besoins sont projetés sur l'ontologie globale de LUBM afin de définir l'ontologie OED. Dans notre scénario, l'ensemble des besoins spécifiés permettent de définir l'ontologie OED comme module de l'ontologie globale (scénario 2 : $OED \subset G$). Cette ontologie est annotée par les concepts multidimensionnels selon l'algorithme 1 d'annotation. L'algorithme a permis d'identifier trois classes Fait : *Publication*, *Student* et *Employee*. Les classes dimensions sont *Institute*, *University*, *Program* et *Research*. La figure 2.14 présente le modèle multidimensionnel obtenu sous forme d'un diagramme de classes.

2.4.2.2 Le processus ETL

Dans cette étape, nous déroulons l'algorithme ETL au niveau ontologique sur l'ontologie OED afin d'alimenter son schéma par les instances des sources. Pour ce faire, il suffit de traduire les opérateurs ETL par des requêtes Sparql, ensuite de de dérouler l'algorithme 4 sur l'OED.

2.4.2.3 Définition de l'ED sémantique final

Cette dernière section présente les phases de modélisation logique et physique. *Oracle* permet de stocker les ontologies en utilisant une représentation *verticale* où tous les triplets sont stockés dans la même table. Les données sémantiques sont stockées dans le schéma *MDSYS*. La BDBO d'Oracle suit une architecture de *type I* où il n'y a pas de séparation entre les différents schémas (ontologiques et de données).

Nous avons procédé à la création de l'ED sémantique en créant une nouvelle BDBO Oracle, dont le schéma logique correspond à la traduction du schéma de l'OED dans une représentation verticale. Cette ontologie est stockée dans un fichier N-triple (défini selon une représentation verticale). L'éditeur d'ontologies Protégé fournit un moyen pour sauvegarder une ontologie OWL sous ce format N-triple. Nous avons ensuite utilisé l'utilitaire SQL*Loader pour le chargement du fichier N-triple dans la nouvelle BDBO. SQL*Loader est un utilitaire d'Oracle qui permet le chargement de données à partir d'un fichier plat (fichier N-triple) vers une table de la base de données Oracle. Comme la démarche le préconise, nous étendons le métaschéma ontologique d'Oracle par le modèle des besoins défini, comme présenté dans le chapitre précédent.

Nous avons ensuite procédé à une série d'expérimentation pour tester notre méthode. Les expérimentations du déroulement du processus ETL sur ce scénario nous ont permis d'étudier trois aspects relatifs [22] : à la *complexité* de l'algorithme proposé, à la *scalabilité* de l'approche, et à l'étude des *performances* du processus ETL. Nous rappelons que très peu de travaux ETL dans la littérature proposent des expérimentations validant le processus ETL, et se contentent de dérouler leurs propositions sur un exemple proposé. Certains travaux proposent des outils supportant le processus ETL proposé. Il n'y a donc aucun consensus établi concernant les critères d'évaluation d'un processus ETL.

Nos expérimentations ont été réalisées sur un ordinateur portable (HP) Intel (R) Core^{TM} i5-3320M CPU 2,60 GHz, avec un disque dur de 500 Go et une RAM de 4 Go. Nous avons utilisé le système d'exploitation *Windows XP* Professional et le *SDK Java 1.7*. Nous commençons par examiner l'espace de calcul nécessaire pour peupler chaque classe de l'ontologie de l'ED. La figure 2.15 présente le nombre d'itérations de l'algorithme ETL proposé, nécessaires pour converger, par rapport au nombre de classes de l'ontologie OED. L'algorithme est basé sur des mappings définis au niveau intentionnel (concepts) et pas au niveau extensionnel (instances). Les résultats montrent que l'espace de calcul reste stable relativement au nombre de classes. Bien que le nombre d'itérations moyen par source soit de 15 ; dans le pire des cas notre algorithme ne calcule pas plus de 125 itérations. Ces résultats vérifient la faisabilité de notre approche.

Nous avons ensuite examiné la *scalabilité* et les *performances* du processus ETL sur l'ED. La figure 2.16 illustre les résultats de l'intégration pour les six BDBO. Pour ces sources, nous calculons le temps nécessaire pour l'intégration des instances des sources au niveau du schéma de l'ED. Nous faisons le test en faisant évoluer le nombre d'instances à chaque test (tableau 2.2). La courbe du temps montre que l'exécution de l'algorithme se fait de manière linéaire par rapport au nombre d'instances utilisées, ce qui signifie que la méthode proposée est évolutive. D'autre part, le temps nécessaire pour intégrer les six BDBO créées en utilisant notre méthode n'excède pas 4 minutes, ce qui indique les performances acceptables de l'algorithme proposé. Nous présentons dans ce qui suit l'outil développé pour supporter la méthode de conception proposée.

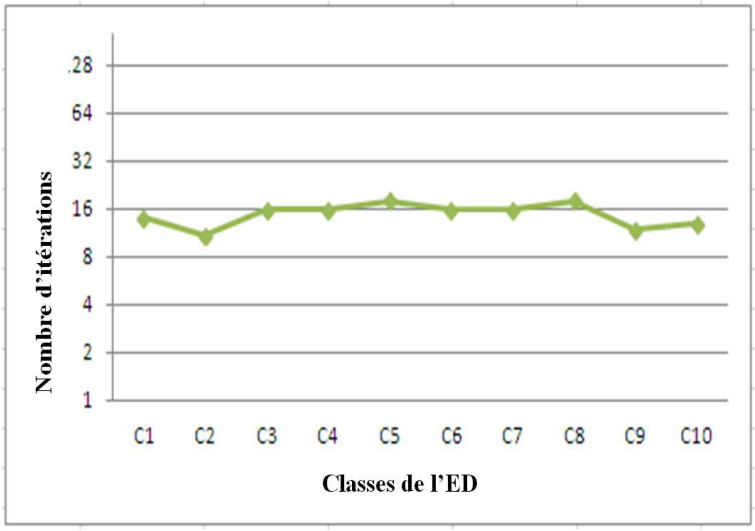

FIGURE 2.15 – Complexité de l'algorithme proposé

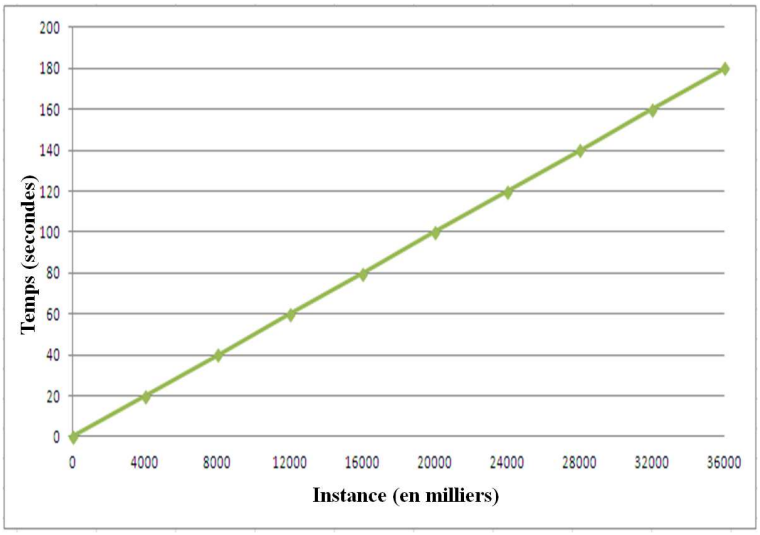

FIGURE 2.16 – Complexité en temps relativement au nombre d'instances générées

2.4.3 Prototype d'outil implémentant la méthode de conception

2.4.3.1 Environnement de développement

Nous proposons un prototype d'outil implémentant toutes les étapes de conception citées (conception du schéma et conception ETL). L'outil fournit une aide à la conception, il est destiné aux concepteurs et administrateurs afin de les accompagner dans le processus de conception de l'ED. L'outil a été développé dans un environnement *Windows XP*, en langage de programmation Java *JDK 1.7*. Les ontologies manipulées par l'outil sont au format *OWL*. L'accès et la manipulation des ontologies se fait via les API de développement *OWLAPI* et *ProtegeOWL*. Le raisonneur utilisé est *Fact++*. L'outil que nous proposons repose sur une ontologie de domaine existante, partagée entre les sources. Chaque BDBO source référence l'ontologie partagée par des assertions de mappings définies.

2.4.3.2 Architecture technique de déploiement d'un EDS

L'architecture technique de l'implémentation de notre solution, illustrée dans la figure 2.17, est définie selon le modèle MVC (Modèle-Vue-Contrôleur) en trois couches :

- la couche *vue* offre les interfaces aux utilisateurs pour : la configuration des paramètres de stockage et de connexion aux BDBO sources, l'affichage et la spécification des besoins, l'affichage et la visualisation des modèle de l'ontologie partagée, de l'ontologie de l'ED et du modèle multidimensionnel.
- la couche *contrôleur* gère les événements parvenant de la couche vue et ceux de la couche modèle.
- la couche *modèle* contient la partie métier qui correspond aux étapes de conception du schéma de l'ED et aux opérateurs ETL encapsulés par des objets DAO qui gèrent les accès aux systèmes de stockages (abstraction et encapsulation des accès).

2.4.3.3 Etapes et modules d'implémentation

Le scénario de démonstration que l'on présente ici implique un ensemble de BDBO dont la sémantique est couverte par l'ontologie partagée du banc d'essai LUBM. Les modules définis et implémentés dans l'outil sont illustrés dans la figure 2.18. Nous présentons dans ce qui suit chacun de ces modules.

2.4.3.3.1 Les sources de données

Le premier module *connexion* présente le module de connexion aux BDBO à intégrer dans l'ED. Les paramètres que l'utilisateur doit spécifier sont les suivants (figure 2.19) :

- *L'identification des sources* : dans cette partie, l'utilisateur saisit un nom et un espace de nom pour chaque BDBO participant au processus de conception.
- *Les paramètres de stockage* : l'utilisateur précise les paramètres de stockage de chaque BDBO, à savoir le type d'architecture de la source (type I, II ou III), le modèle de stockage supporté par la source (schéma horizontal, triplet, binaire, etc) et le SGBD utilisé par la source.

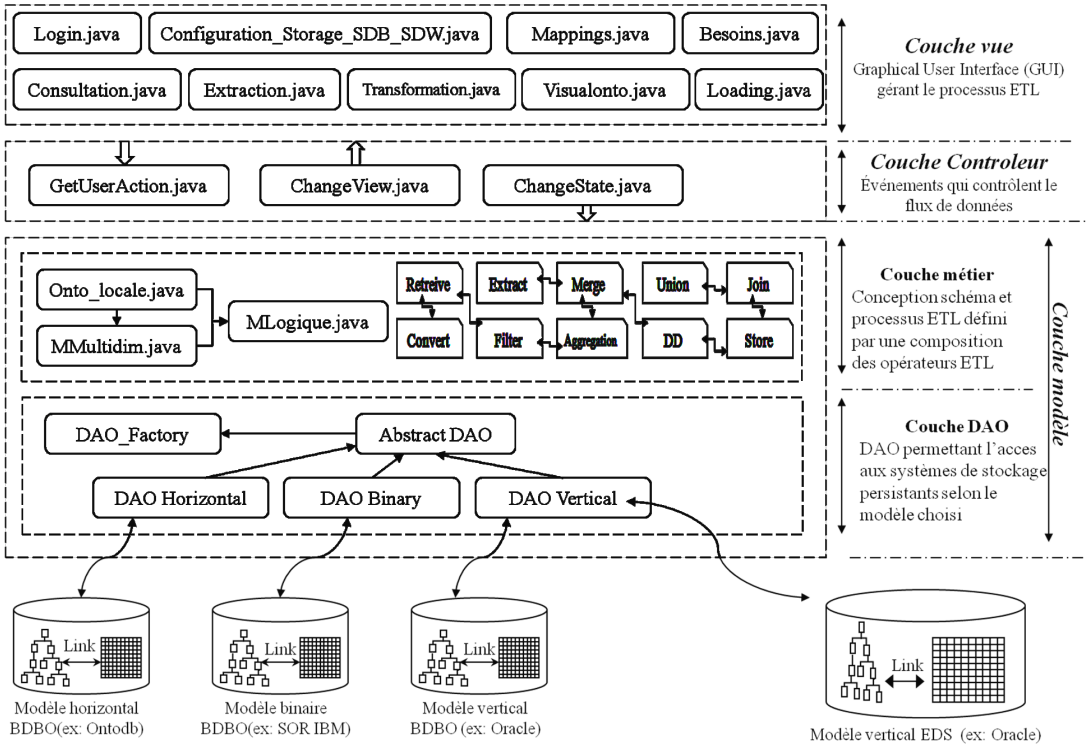

FIGURE 2.17 – Architecture technique de l'outil de conception de l'ED sémantique

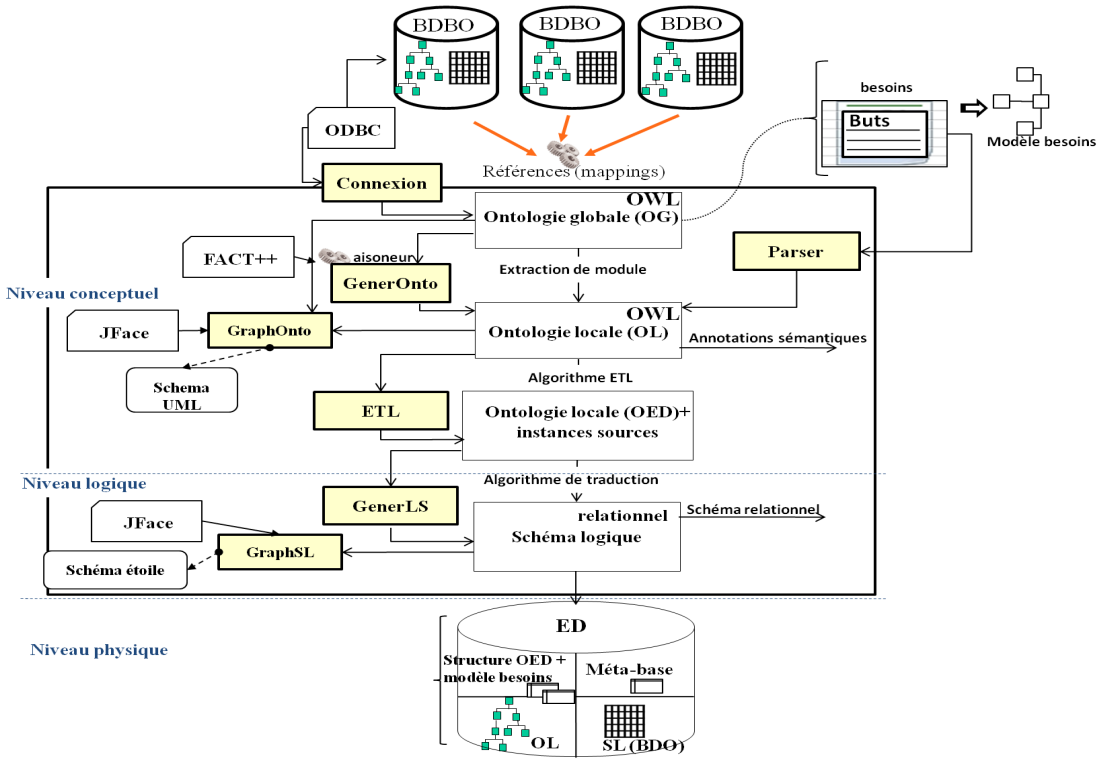

FIGURE 2.18 – Modules d'implémentation de l'outil

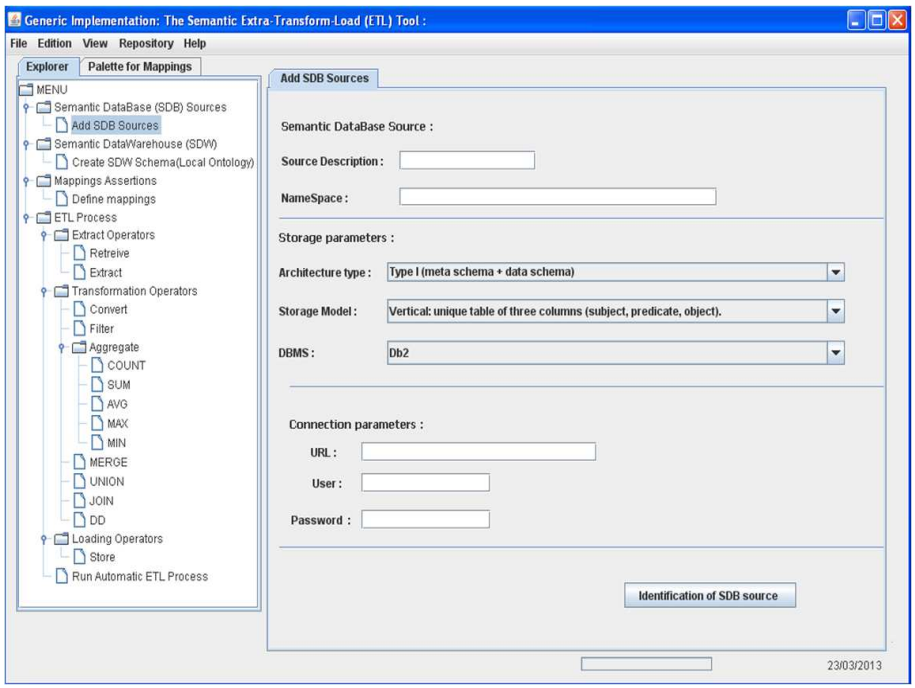

FIGURE 2.19 – Paramètres d'identification et de connexion aux sources

– *Les paramètres de connexion aux sources* : l'utilisateur spécifie le driver et l'URL de la source, ainsi que le nom d'utilisateur et le mot de passe (si nécessaire). La connexion aux sources se fait via le gestionnaire ODBC. Une fois ces paramètres définis, l'outil procède au chargement de la source.

2.4.3.3.2 Le schéma global

Le schéma global est représenté par l'ontologie de domaine partagée. L'accès à l'ontologie s'effectue via l'API de développement java *Protégé-OWL* fournie par l'éditeur d'ontologies Protégé.

2.4.3.3.3 Les mappings

Les mappings entre les schémas des sources et le schéma global sont définis selon les quatre types de mappings identifiés. Pour chaque classe source, l'utilisateur (concepteur) définit le mapping de la source avec les classes et propriétés de l'ontologie partagée. Ces mappings sont définis en se basant sur les constructeurs de LD (figure 2.20).

L'outil permet de tester chaque opérateur ETL et d'exécuter le processus ETL correspondant de façon automatique par une requête Sparql (figure 2.21).

2.4.3.3.4 Visualisation des ontologies

Le module *GraphOnto* de l'outil offre deux types de visualisation des ontologies (partagée et OED) : (1) sous forme d'une arborescence, en parcourant les classes de l'ontologie et en les affichant sous forme d'une hiérarchie (figure 2.22). L'ensemble des classes, leurs descriptions, leurs propriétés et leurs individus sont présentés. (2) Une visualisation graphique sous forme d'un diagramme de classes. Cette visualisation est implémentée en utilisant la bibliothèque graphique

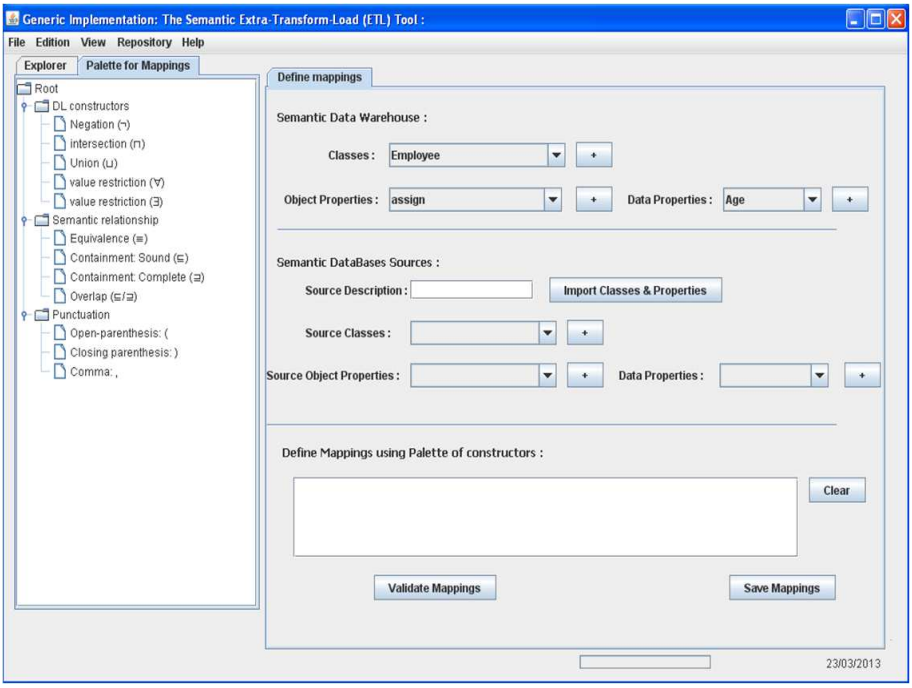

FIGURE 2.20 – Interface de définition des mappings entre les BDBO sources et l'ontologie

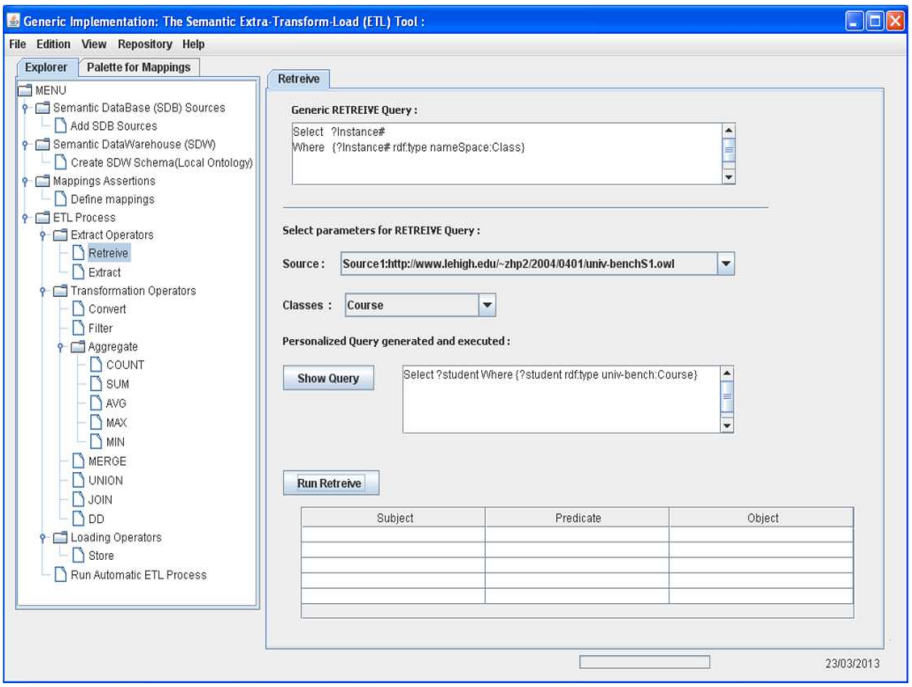

FIGURE 2.21 – Interface d'exécution des opérateurs ETL

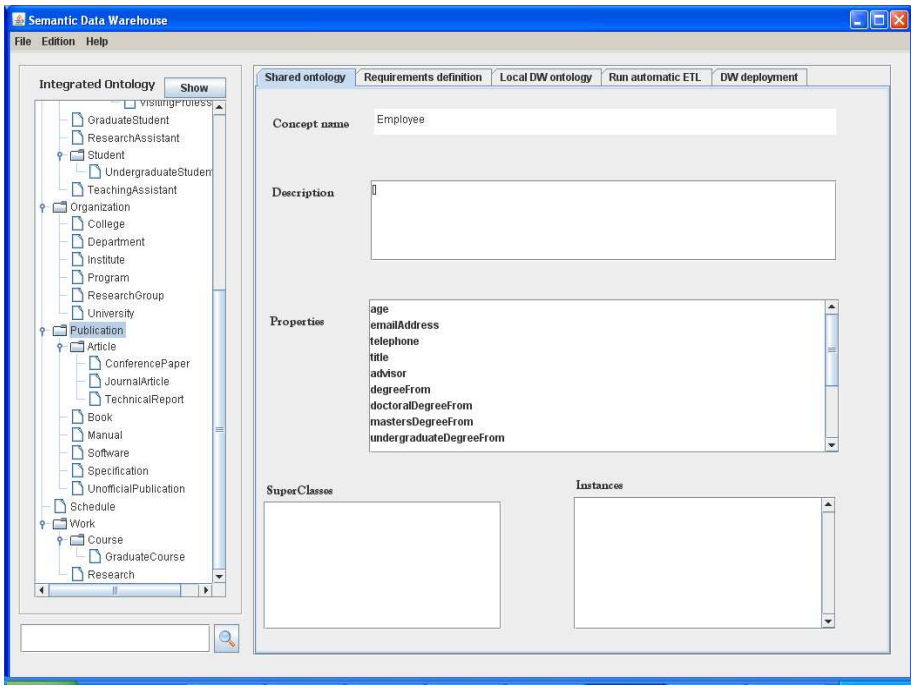

FIGURE 2.22 – Interface de visualisation de l'ontologie sous forme d'une arborescence de classes

libre *Java JFace* (figure 2.23).

2.4.3.3.5 Définition des besoins

Les besoins décisionnels sont exprimés sur l'ontologie partagée, en utilisant le modèle de besoins orienté but proposé. Une interface est fournie pour faciliter à l'utilisateur de spécifier ses besoins (figure 2.24). Pour chaque but défini, l'utilisateur sélectionne les cordonnées du but (résultat et critères) en choisissant les concepts et propriétés pertinents de l'ontologie partagée. Une fois les buts validés, le module *parseur* permet d'analyser l'ensemble des buts afin d'extraire l'ontologie locale à l'ED.

2.4.3.3.6 La phase conceptuelle

La phase conceptuelle comprend la définition de l'ontologie locale OED et la phase de conception ETL.

La définition de l'OED

L'ontologie locale à l'ED est générée à partir de l'ontologie partagée en projetant les besoins des décideurs et utilisateurs sur l'ontologie. Le module *GeneOnto* permet de générer cette ontologie locale qui représente un module fragment de l'ontologie partagée ; représentant les informations pertinentes des sources que l'ED devra représenter. L'extraction de l'ontologie OED s'effectue en utilisant le plugin de modularité *ProSé* disponible sous l'éditeur d'ontologies *Protégé* et accessible via l'*API OWL*. Le plugin *ProSé* permet l'extension de l'OED par de nouveaux concepts et propriétés au cas où les besoins définis ne couvrent pas tous les aspects applicatifs, tout en assurant la cohérence et complétude de l'ontologie définie.

Les mécanismes de raisonnement utilisés par la méthode sont implémentés en utilisant le

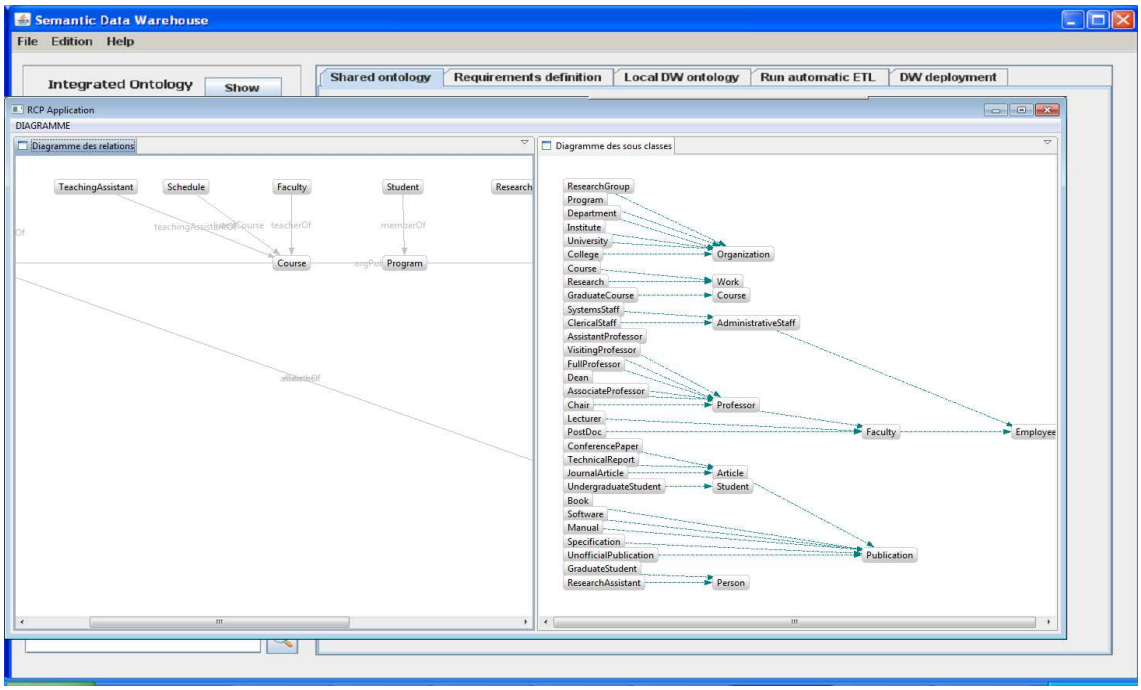

FIGURE 2.23 – Interface de visualisation de l'ontologie sous forme graphique

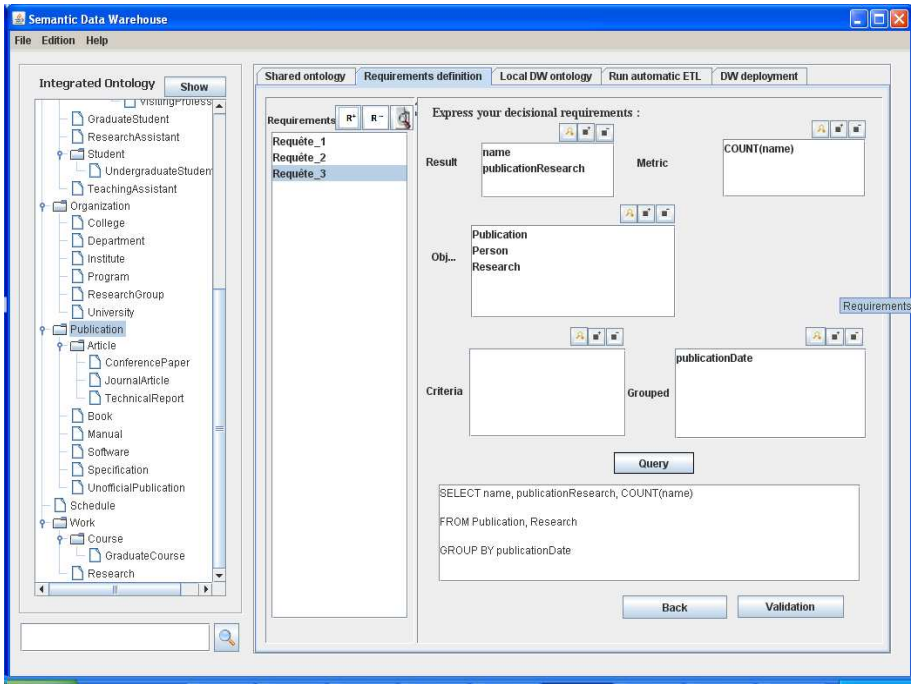

FIGURE 2.24 – Interface de spécification des besoins des utilisateurs

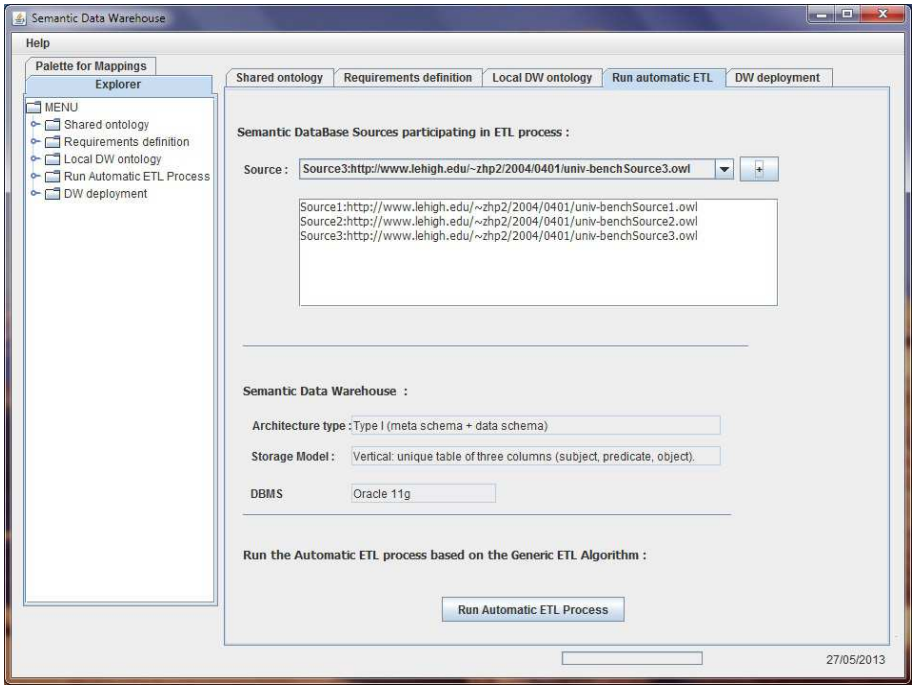

FIGURE 2.25 – Interface d'exécution du processus ETL

raisonneur *Fact++* [14]. Les règles de propagation des relations d'influence entre les buts sont définies par des règles *SWRL* [15] (*semantic web rule language*) et exécutées par le moteur d'inférence *Jess* [16].

L'outil permet la visualisation de l'ontologie OED de la même manière que l'ontologie partagée (arborescence et diagramme de classes). L'algorithme d'annotations des concepts multidimensionnels est implémenté, et permet d'annoter les concepts et propriétés de l'ontologie OED par leur rôle multidimensionnel.

La conception ETL au niveau ontologique

Une principale contribution de notre méthode est de définir la phase ETL au niveau conceptuel. L'algorithme ETL proposé a été implémenté dans le module *ETL*. Sur la base des mappings existants entre les schémas des BDBO sources et le schéma de l'ED, l'outil permet une extraction automatique des instances ontologiques provenant des sources, leur transformation (filtrage, conversion et agrégation) et le calcul des nouvelles valeurs respectant le schéma de l'ED. Pour ce faire, le module ETL définit l'instanciation des opérateurs ETL selon le langage d'interrogation cible (Sparql), et procède au chargement des données transformées dans les classes appropriées du modèle de l'OED (figure 2.25).

2.4.3.3.7 La phase de conception logique

Le module *GenerSL* implémente la phase de modélisation logique. Le schéma relationnel, représenté sous forme d'un schéma en étoile, est défini par application des règles de traduction

14. http ://owl.man.ac.uk/factplusplus/
15. http ://www.w3.org/Submission/SWRL/
16. http ://www.jessrules.com/

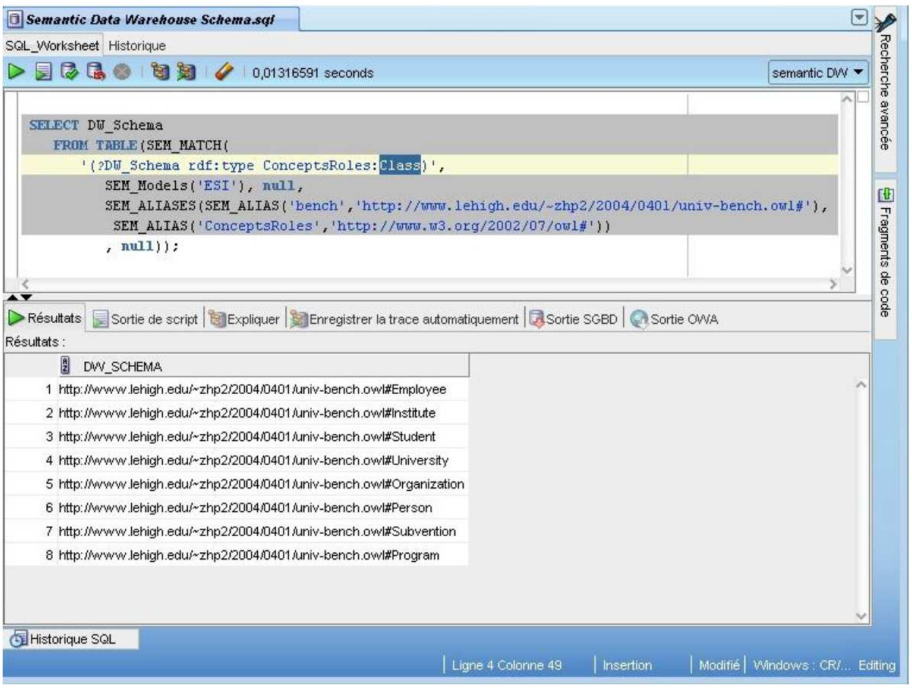

FIGURE 2.26 – Déploiement de l'ED final

de l'ontologie OWL en schéma relationnel. Ce modèle est représenté sous forme graphique.

2.4.3.3.8 La phase de conception physique et déploiement

Le modèle et instances intégrées de l'OED sont représentées dans l'ED final selon le schéma du SGBD cible. La figure 2.26 montre le schéma de l'ED cible implémenté sous le SGBD sémantique d'*Oracle*.

2.5 Conclusion

Nous avons proposé dans ce chapitre une méthode de conception et d'alimentation d'un ED en revisitant l'ordre d'exécution de la phase ETL. Le principal constat ayant motivé cette proposition provient du fait que la plupart des méthodes de la littérature fournissent à l'ED un schéma unique de déploiement. L'étude de l'évolution du cycle de conception des ED nous a fait constater la multiplicité des schémas et des architectures de déploiement. Notre proposition poursuit le même objectif principal énoncé en introduction, qui est de supporter les dernières évolutions du cycle de conception. Nous avons donc commencé par fournir un Framework d'intégration <G,S,M> basé sur le formalisme des LD et représentant une intégration au niveau conceptuel. Dans notre méthode de conception, l'ontologie partagée par les sources joue le rôle du schéma global dans le Framework <G,S,M>. Les sources sont représentées aussi bien par leurs données que par leur modèle conceptuel. Les mappings sont définis entre le schéma conceptuel global et les schémas conceptuels des sources. La méthode que nous proposons dans ce chapitre complète la méthode présentée au chapitre précédent permettant de définir le schéma de l'ED. Les besoins des utilisateurs sont spécifiés selon un modèle pivot orienté but, en utilisant les concepts et propriétés

du schéma global (l'ontologie partagée), afin de déterminer le schéma conceptuel de l'ED. Le processus ETL est introduit dès la phase conceptuelle au niveau ontologique. Le processus ETL est basé sur un ensemble d'opérateurs ETL conceptuels et un algorithme ETL qui se base sur ces opérateurs ETL et sur différents types de mapping afin d'alimenter le schéma de l'ED. Comme le schéma de l'ED est représenté par l'ontologie locale, nous avons fourni la traduction des opérateurs ETL conceptuels en des requêtes ontologiques Sparql. Le schéma de l'ED peut être traduit en un modèle logique selon différentes plateformes de déploiement disponibles. Le modèle final de l'ED est stocké dans une BDBO, représentant le modèle de besoins et les modèles de données. La méthode proposée couvre et formalise toutes les étapes du cycle de conception de l'ED.

Afin de valider notre proposition, nous avons appliqué la méthode proposée en instanciant le Framework d'intégration par des sources de type BDBO référençant une ontologie de domaine partagée. Ces sources sont d'excellentes candidates pour une intégration complètement conceptuelle puisqu'elles contiennent leur schéma conceptuel sous forme d'une ontologie locale à la source. De plus, cette validation permet de répondre à notre deuxième objectif qui est de fournir une méthode permettant de concevoir un ED à partir des BDBO qui deviennent des sources potentielles disponibles au sein des entreprises. Pour ce faire, nous avons présenté une étude de cas utilisant le banc d'essai *LUBM* et le SGBD sémantique industriel *Oracle*. Les expérimentations menées ont permis d'étudier trois aspects de la méthode relatifs à la complexité de l'algorithme ETL proposé, à la scalabilité de l'approche, et aux performances du processus ETL. Les résultats montrent que l'espace de calcul de l'algorithme ETL est stable relativement au nombre de classes. La deuxième série d'expérimentations teste la scalabilité et les performances du processus ETL lorsque nous faisons évoluer le nombre d'instances à intégrer. La courbe du temps montre que l'exécution de l'algorithme se fait de manière linéaire par rapport au nombre d'instances, ce qui montre la scalabilité du processus ETL. Pour compléter notre validation, nous avons fourni un outil implémentant l'ensemble des étapes de la méthode proposée.

Pour récapituler, nous notons que nous avons fourni à notre connaissance, la première méthode de conception complète qui couvre les cinq étapes du cycle de conception des ED. L'ED défini représente aussi bien le modèle de données que le modèle de besoins au sein de sa structure. Ce modèle de l'ED est alimenté par les données issues des sources via la définition d'un processus ETL s'effectuant dès la phase conceptuelle. Ceci afin de permettre un déploiement *à la carte* de l'ED.

Troisième partie

Conclusion et perspectives

Conclusion et perspectives

Conclusion

Nous présentons dans ce chapitre un bilan du travail que nous avons effectué ainsi qu'un ensemble d'ouvertures et de perspectives pour ce travail.

Les principales propositions de cette thèse découlent d'un important constat : le cycle de conception des applications d'ED a évolué, mais les méthodes de conception proposées ne s'adaptent pas à cette évolution. Notre étude de l'état de l'art relatif aux projets d'entreposage nous a fait remarquer que les différentes méthodes proposées étudient les phases de conception de manière séparée, et fournissent très rarement une vision globale de conception de l'ED englobant toutes les phases de conception. Les différentes contributions proposées dans ces travaux ont une portée limitée au sein d'une phase de conception, et ne s'inscrivent pas dans une vision orientée cycle de conception.

Différentes évolutions ont mené à l'établissement du cycle de conception des applications de gestion des données de manière générale, et à son adoption par la communauté. Nous avons distingué trois principales évolutions : une évolution *verticale* donnant lieu aux différentes phases de conception uniformisées par l'architecture ANSI/SPARC, une évolution *interne* donnant lieu à l'établissement des étapes de chaque phase, et une évolution *horizontale* donnant lieu à l'enrichissement de chaque phase par différents modèles de données et architectures. Ces évolutions ont d'abord été remarquées pour les cas des applications de BD. Les applications d'ED ont consolidé ce cycle de conception et l'ont enrichi de manière *horizontale* par l'apparition de nouveaux modèles de données ; et de manière *verticale* par l'ajout d'une nouvelle phase d'extraction, transformation et chargement des données (ETL). Ces évolutions offrent aux acteurs du système comme les concepteurs et les administrateurs, plusieurs choix de conception selon différents modèles de stockage, sur diverses plateformes de déploiement logiques et physiques. Une méthode de conception doit accompagner les concepteurs tout au long du processus de conception en leur permettant d'effectuer le choix le plus pertinent pour leurs systèmes et applications.

Contributions, Expérimentations et Outillage

Les différentes contributions de notre thèse sont les suivantes :

Etats de l'art

Nous avons présenté une synthèse des principales évolutions ayant donné lieu au cycle de conception des applications de gestion des données : les BD et ensuite les ED. Cette partie nous

a permis d'établir les principales évolutions qu'a connu le cycle de conception et d'argumenter les motivations ayant mené à ces évolutions. Nous avons fait remarquer que l'accès à la sémantique a souvent été une motivation pour différentes propositions d'enrichissement du cycle de conception. Nous avons ensuite proposé une synthèse détaillée ainsi qu'une classification des principaux travaux de conception du schéma de l'ED, et des travaux de conception ETL. Ces travaux sont analysés selon différents critères de comparaison qui nous ont permis de situer nos contributions.

Le cœur de notre travail a été présenté dans la deuxième partie de ce manuscrit que nous avons organisé en deux principales propositions.

Persistance des besoins pour un système d'ED manipulant données et traitements

Notre première proposition consolide le cycle de conception des ED en donnant la possibilité de simuler les différents choix de conception qui s'offrent aux concepteurs. Puisque toute architecture technique d'un système découle toujours des besoins fonctionnels et non fonctionnels collectés, nous avons proposé de fournir une vue persistante des besoins des utilisateurs au sein de la structure finale de l'ED.

Pour cette première proposition, plusieurs contributions sont présentées. Une première méthode de conception du schéma de l'ED, basée sur une ontologie de domaine, spécifiant les sources de données et les besoins des utilisateurs au niveau ontologique. Nous avons mis en avant la valeur ajoutée de l'utilisation des ontologies pour la spécification et la validation des besoins des utilisateurs.

La méthode de conception prend en entrée une ontologie de domaine partagée par les sources de données, et un ensemble des besoins des utilisateurs définis selon une approche orientée buts. En appliquant un ensemble d'étapes de conception suivant les niveaux de l'architecture ANSI/SPARC, la méthode fournit le schéma conceptuel ontologique de l'ED ainsi que son schéma physique. Nous avons proposé de représenter l'ED dans une architecture faisant cohabiter les deux principaux modèles de l'ED : le modèle de données (conceptuel, logique et physique) ainsi que le modèle de besoins. Nous avons pour ce faire utilisé une architecture de BDBO. Une validation montrant la faisabilité de l'approche a été proposée sur deux architectures de BDBO.

Nous avons montré de manière théorique ensuite par des expérimentations l'intérêt de faire persister les besoins dans l'ED, notamment pour : (1) simuler certaines tâches de conception dès le début du projet d'entreposage, (2) renforcer les liens entre les différentes phases de conception de l'ED et faciliter la communication entre les acteurs (administrateur, concepteur, éditeur, etc) intervenant tout au long du cycle de conception et (3) identifier les rôles clés des acteurs du système en apportant un modèle économique dans la conception des ED plus adapté au contexte actuel, basé sur les acteurs pertinents. Les expérimentations ont été menées en utilisant le banc d'essai SSB, elles permettent de simuler différentes tâches d'optimisation de l'ED, à partir des besoins des utilisateurs.

ETL au niveau ontologique pour un déploiement à la carte

Notre deuxième proposition consolide le cycle de conception des ED en offrant un déploiement logique "à la carte" de l'ED selon les besoins des concepteurs. Nous avons identifié que la principale limite à un déploiement à la carte, consiste dans l'ordre d'exécution de la phase ETL

habituellement placée suite à la phase de modélisation logique. Pour répondre à notre objectif, nous fournissons une méthode de conception complétant la première méthode par la phase ETL exécutée dès la phase conceptuelle. Notre méthode est plus adaptée au cycle de conception actuel et répond au différents choix de déploiement possibles connus après l'évolution horizontale du cycle de conception.

Pour notre seconde proposition, trois principales contributions sont présentées. Nous avons commencé par proposer un Framework d'intégration générique, prenant en compte les différents modèles ontologiques et la diversité des sources candidates aux projets d'entreposage. Nous avons fourni un algorithme ETL basé sur des opérateurs conceptuels. Nous avons ensuite présenté la méthode de conception englobant toutes les phases du cycle de conception et exécutant le processus ETL durant la phase conceptuelle, plus précisément au niveau ontologique.

Afin de valider notre approche, nous avons instancié le Framework d'intégration et la méthode proposée pour la conception d'un ED à partir des BDBO. Les BDBO sont actuellement adoptées par les deux communautés industrielle et de recherche. Cette instanciation permet d'une part de tester et de valider notre approche. D'autre part, elle permet de fournir un moyen de construire un ED à partir des sources de type BDBO qui deviennent des sources candidates sérieuses pour les projets d'entreposage. Des expérimentations ont été conduites, utilisant le banc d'essai du domaine universitaire LUBM, et permettent d'évaluer les performances de la méthode proposée.

Un prototype d'outil implémentant l'ensemble des étapes de la méthode est fourni, permettant de faciliter les différentes tâches de conception identifiées et de guider le concepteur tout au long du processus de conception.

Perspectives

Les travaux présentés dans ce manuscrit laissent envisager de nombreuses perspectives. Dans cette section, nous présentons succinctement celles qui nous paraissent être les plus importantes.

Etude de l'évolution des besoins des utilisateurs et des ontologies manipulées

Notre première proposition a consisté à fournir une vue persistante des besoins des utilisateurs au sein de l'ED, au terme d'une méthode de conception ontologique. L'étude de l'évolution des besoins des utilisateurs est un aspect important pour assurer la faisabilité de l'approche dans un cas réel d'un système d'ED développé au sein d'une entreprise ou d'une organisation. Nous pensons que l'étude de l'évolution des besoins dans le cadre des applications de BD classiques et leur adaptation aux applications d'ED, pourrait être une piste à explorer. Nous pensons également à lier le modèle de besoins avec le modèle ETL, où chaque processus ETL sera identifié par un besoin fonctionnel donné. Nous avons proposé un premier modèle répondant à cet objectif dans [22]. Cette proposition permettra de définir le processus ETL nécessaire pour la satisfaction de chaque besoin. L'évolution des besoins pourra ainsi être mieux gérée, dans le sens où les données impactées par le changement seront identifiées. L'étude de l'évolution des ontologies manipulées par la méthode proposée est également un aspect à étudier pour la faisabilité de l'approche. Cette évolution peut concerner différents aspects comme l'évolution du schéma de l'ontologie ou l'évolution des instances ontologiques. Nous posons cependant l'hypothèse raisonnable que l'ontologie de domaine évolue lentement, ce qui présente un aspect intéressant de notre méthode.

Proposition d'un modèle de besoins complet et générique

Edgar Codd [50] définit un modèle de données par trois principales composantes : la partie statique *structurelle* du modèle, la partie dynamique de *manipulation* du modèle, et la partie *intégrité* du modèle. En proposant le modèle de besoins, nous avons fourni la partie structurelle du modèle. Il est important de fournir la partie manipulation en proposant un langage de manipulation et d'interrogation dédié aux besoins des utilisateurs, qui serait l'équivalent du SQL pour les données. Cette étude permettra de faciliter la gestion des besoins.

Proposition d'un simulateur supportant le cycle de vie d'un système d'ED

L'objectif de la première proposition est de fournir un moyen de simuler le processus de conception de l'ED dès le début du projet. Nous avons fourni via des expérimentations le moyen de simuler certaines tâches de conception comme l'optimisation de l'ED. Il est intéressant de proposer un simulateur complet supportant toutes les phases de conception de l'ED afin de faciliter aux concepteurs et aux administrateurs une prise de décision pertinente par rapport aux choix de conception (conceptuels, logiques et physiques) qui s'offrent à eux. Cette étude fait actuellement l'objet d'une thèse en cours de réalisation au niveau du laboratoire LIAS.

Validation et étude de critères de qualité

La dernière perspective tend à valider nos propositions par une validation empirique sur des études de cas réels. Nos propositions pourraient être validées sur d'autres types de BD et plateformes de déploiement logiques et physiques. De nouveaux scénarios d'intégration complétant le scénario présenté peuvent consolider la validation. Le scénario que nous avons considéré est celui d'un schéma directeur, où les sources s'accordent à référencer le schéma imposé par l'organisme directeur, sans apporter d'extension locale. Un scénario à étudier est le scénario d'existence d'extensions locales et leur prise en compte lors de l'intégration.

Cette validation doit être complétée par l'étude des critères de qualité d'une méthode de conception d'un ED. Ce thème fait l'objet de recherches actives. Les résultats de nos expérimentations pourront ainsi être évalués selon des modèles de qualités consensuels aussi bien pour la conception du schéma de l'ED que pour la phase d'alimentation de l'ED.

Quatrième partie

Annexes

Annexe A : Besoins et requêtes du banc d'essai SSB

L'annexe suivante présente l'ensemble des besoins décisionnels et des requêtes fournies par le banc d'essai Star Schema Benchmark (SSB) [17].

Les trois premières requêtes de la spécification décrivent le premier besoin décisionnel. Ce dernier a pour objectif de mesurer l'évolution ou l'augmentation des recettes selon les remises accordées, pour certains produits dans une période donnée. Nous présentons les trois premières requêtes Q1.1, Q1.2 et Q1.3.

Q1.1

```
select sum(lo_extendedprice*lo_discount) as revenue
from
    lineorder, dates
where
    lo_orderdate = d_datekey
    and d_year = 1993
    and lo_discount between 1 and 3
    and lo_quantity < 25;
```

Q1.2

```
select
    sum(lo_extendedprice*lo_discount) as revenue
from
    lineorder, dates
where
    lo_orderdate = d_datekey
    and d_yearmonthnum = 199401
    and lo_discount between 4 and 6
    and lo_quantity between 26 and 35;
```

17. http ://www.cs.umb.edu/ poneil/StarSchemaB.PDF

Q1.3

```
select
    sum(lo_extendedprice*lo_discount) as revenue
from
    lineorder, dates
where
    lo_orderdate = d_datekey
    and d_weeknuminyear = 6
    and d_year = 1994
    and lo_discount between 5 and 7
    and lo_quantity between 26 and 35;
```

Les trois requêtes suivantes décrivent le deuxième besoin décisionnel. Ce besoin compare les recettes selon les fournisseurs dans une région donnée pour certaines classes de produits et par année. Nous présentons les trois requêtes Q2.1, Q2.2 et Q2.3.

Q2.1

```
select
    sum(lo_revenue), d_year, p_brand
from
    lineorder, dates, part, supplier
where
    lo_orderdate = d_datekey
    and lo_partkey = p_partkey
    and lo_suppkey = s_suppkey
    and p_category = 'MFGR#12'
    and s_region = 'AMERICA'
group by
    d_year, p_brand
order by
    d_year, p_brand;
```

Q2.2

```
select
    sum(lo_revenue), d_year, p_brand
from
    lineorder, dates, part, supplier
where
    lo_orderdate = d_datekey
    and lo_partkey = p_partkey
    and lo_suppkey = s_suppkey
    and p_brand between 'MFGR#2221' and 'MFGR#2228'
```

```
    and s_region = 'ASIA'
group by
    d_year, p_brand
order by
    d_year, p_brand;
```

Q2.3

```
select
    sum(lo_revenue), d_year, p_brand
from
    lineorder, dates, part, supplier
where
    lo_orderdate = d_datekey
    and lo_partkey = p_partkey
    and lo_suppkey = s_suppkey
    and p_brand = 'MFGR#2239'
    and s_region = 'EUROPE'
group by
    d_year, p_brand
order by
    d_year, p_brand;
```

Les quatre requêtes suivantes décrivent le troisième besoin décisionnel. Ce besoin mesure les recettes totales des transactions pour une région donnée à une certaine période de temps, selon la provenance des clients et des fournisseurs. Nous présentons les quatre requêtes Q3.1, Q3.2, Q3.3 et Q3.4.

Q3.1

```
select
    c_nation, s_nation, d_year,
    sum(lo_revenue) as revenue
from
    customer, lineorder, supplier, dates
where
    lo_custkey = c_custkey
    and lo_suppkey = s_suppkey
    and lo_orderdate = d_datekey
    and c_region = 'ASIA'
    and s_region = 'ASIA'
    and d_year >= 1992 and d_year <= 1997
group by
    c_nation, s_nation, d_year
order by
```

```
    d_year asc, revenue desc;
```

Q3.2

```
select
    c_city, s_city, d_year, sum(lo_revenue) as revenue
from
    customer, lineorder, supplier, dates
where
    lo_custkey = c_custkey
    and lo_suppkey = s_suppkey
    and lo_orderdate = d_datekey
    and c_nation = 'UNITED STATES'
    and s_nation = 'UNITED STATES'
    and d_year >= 1992 and d_year <= 1997
group by
    c_city, s_city, d_year
order by
    d_year asc, revenue desc;
```

Q3.3

```
select
    c_city, s_city, d_year, sum(lo_revenue) as revenue
from
    customer, lineorder, supplier, dates
where
    lo_custkey = c_custkey
    and lo_suppkey = s_suppkey
    and lo_orderdate = d_datekey
    and (c_city='UNITED KI1' or c_city='UNITED KI5')
    and (s_city='UNITED KI1' or s_city='UNITED KI5')
    and d_year >= 1992 and d_year <= 1997
group by
    c_city, s_city, d_year
order by
    d_year asc, revenue desc;
```

Q3.4

```
select
    c_city, s_city, d_year, sum(lo_revenue) as revenue
from
    customer, lineorder, supplier, dates
where
```

```
    lo_custkey = c_custkey
    and lo_suppkey = s_suppkey
    and lo_orderdate = d_datekey
    and (c_city='UNITED KI1' or c_city='UNITED KI5')
    and (s_city='UNITED KI1' or s_city='UNITED KI5')
    and d_yearmonth = 'Dec1997'
group by
    c_city, s_city, d_year
order by
    d_year asc, revenue desc;
```

Les trois dernières requêtes décrivent le quatrième besoin décisionnel. Ce dernier est un besoin prévisionnel qui vise à calculer le profit moyen dans certaines régions (des clients et des fournisseurs), pour certaines classes de produits et pour une période donnée. Nous présentons les trois requêtes Q4.1, Q4.2 et Q4.3.

Q4.1

```
select
    d_year, c_nation,
    sum(lo_revenue - lo_supplycost) as profit
from
    DATES, CUSTOMER, SUPPLIER, PART, LINEORDER
where
    lo_custkey = c_custkey
    and lo_suppkey = s_suppkey
    and lo_partkey = p_partkey
    and lo_orderdate = d_datekey
    and c_region = 'AMERICA'
    and s_region = 'AMERICA'
    and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by
    d_year, c_nation
order by
    d_year, c_nation;
```

Q4.2

```
select
    d_year, s_nation, p_category,
    sum(lo_revenue - lo_supplycost) as profit
from
    DATES, CUSTOMER, SUPPLIER, PART, LINEORDER
where
    lo_custkey = c_custkey
```

```
    and lo_suppkey = s_suppkey
    and lo_partkey = p_partkey
    and lo_orderdate = d_datekey
    and c_region = 'AMERICA'
    and s_region = 'AMERICA'
    and (d_year = 1997 or d_year = 1998)
    and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by
    d_year, s_nation, p_category
order by
    d_year, s_nation, p_category;
```

Q4.3

```
select
    d_year, s_city, p_brand,
    sum(lo_revenue - lo_supplycost) as profit
from
    DATES, CUSTOMER, SUPPLIER, PART, LINEORDER
where
    lo_custkey = c_custkey
    and lo_suppkey = s_suppkey
    and lo_partkey = p_partkey
    and lo_orderdate = d_datekey
    and s_nation = 'UNITED STATES'
    and (d_year = 1997 or d_year = 1998)
    and p_category = 'MFGR#14'
group by
    d_year, s_city, p_brand
order by
    d_year, s_city, p_brand;
```

Annexe B : Liste des publications

L'annexe suivante présente la liste des publications réalisées dans le cadre de cette thèse. La liste des publications est présentée en deux parties. Chaque partie correspondant à une principale proposition, et comporte la liste des articles acceptés (leur résumé et leur principale contribution), ainsi que la liste des publications en revue acceptées.

Liste des articles relatifs à la première proposition

- Selma KHOURI, Ladjel BELLATRECHE, A Methodology and Tool for Conceptual Designing a Data Warehouse from Ontology-based Sources, ACM Thirteenth International Workshop On Data Warehousing and OLAP (DOLAP 2010), Toronto, Canada, edited by ACM, October, 2010, pp. 19-23. [106].

Résumé. Ontologies become more popular in various domains. In database, they contribute largely in designing operational databases, that we call ontology-based databases (OBDB). An OBDB stores ontology and their instances in the same repository. Several DBMS propose solutions to manipulate and query this type of databases. As consequence, these operational OBDB become a candidate to feed data ware-houses (DW). In this paper, we propose a new conceptual design methodology of DW from data residing in various OBDB that takes into account sources and decision maker requirements. As the result, a semantic DW model is generated. The presence of ontology within a DW facilitates its querying in semantic level and its future integration with other DW built using our methodology. Finally, we present a case tool, called S^2RWC, supporting our proposal.

Objectif de l'article : la principale contribution de cet article est de spécifier les besoins des utilisateurs de l'ED au niveau ontologique.

- Selma KHOURI, Ladjel BELLATRECHE, OntoStar : Outil de conception multidimensionnelle d'un entrepôt de données à base ontologique, 26èmes journées Bases de Données Avancées (BDA'2010), October, 2010. [96].

Résumé. Le développement des outils de conception des entrepôts de données n'a pas eu le même intérêt que celui dans les bases de données traditionnelles. Concevoir un entrepôt de données (ED) est reconnu actuellement comme une tâche cruciale sur laquelle peut reposer la réussite d'un projet décisionnel. Notons que la construction d'un ED peut être assimilée à un système d'intégration (SI) auquel s'ajoute une couche multidimensionnelle. Les ontologies conceptuelles ont largement contribué à la construction des SI. Avec l'émergence des ontologies dans divers do-

maines, un nombre important de sources référençant des ontologies ont vu le jour et sont souvent impliquées dans la construction d'un ED. Dans ce papier, nous proposons un outil graphique assistant les concepteurs des ED. Il repose sur une méthode de conception multidimensionnelle basée sur une ontologie OWL intégrant les différentes sources ontologiques, sur laquelle un ensemble de besoins décisionnels est exprimé.
Objectif de l'article : la contribution de cette article démonstration est de présenter l'outil implémentant la méthode de l'article précédent, permettant la conception du schéma de l'ED spécifiant les sources et les besoins au niveau ontologique. La conférence BDA est une conférence francophone majeure dans le domaine des bases de données, dont le but est de présenter les articles des chercheurs pour la communauté francophone, même si l'article a déjà été publié dans une autre conférence (pas de proceeding publié). Les articles retenus sont ceux évalués positivement par un comité de lecture.

- Selma KHOURI, Ladjel BELLATRECHE, DWOBS : Data Warehouse Design from Ontology-based Sources, in 16th International Conference on Database Systems for Advanced Applications (DASFAA' 2011), April, 2011, pp. 438-441. [97].

Résumé. In the past decades, data warehouse (DW) applications were built from traditional data sources. The availability of domain ontologies creates the opportunity for sources to explicit their semantics and to exploit them in many applications, including in data warehousing. In this paper, we present DWOBS, a case tool for ontological-based DW design based on domain ontologies. It takes as inputs (i) a set of sources referencing OWL ontology and (ii) a set of decisional requirements formulated using Sparql syntax on that ontology. DWOBS gives a semantic multidimensional model of the DW to be designed.
Objectif de l'article : la contribution de cette article démonstration est de présenter l'outil implémentant la méthode de conception du schéma de l'ED spécifiant les sources et les besoins au niveau ontologique. Comparativement à l'outil présenté dans le précédent article, des fonctionnalités additionnelles ont été implémentées dans cet outil comme le raisonnement ontologique.

- Selma KHOURI, Ladjel BELLATRECHE, Osons tout Persister dans un Entrepôt : Données et Modèles, 7èmes Journées Francophones sur les Entrepôts de Données et analyse en Ligne (EDA'11), edited by RNTI, 2011. [98].

Résumé. La structure de stockage actuelle des entrepôts de données matérialise des données conformément à une modèle logique défini par les concepteurs. Récemment, plusieurs travaux ont montré l'intérêt de la modélisation conceptuelle dans le contexte des entrepôts de données du fait qu'elle offre une abstraction du domaine étudié. Sa disponibilité facilite l'interrogation de l'entrepôt de données, car il est plus riche que le modèle logique. Les modèles conceptuel et logique de l'entrepôt sont définis à partir de deux composantes : les sources de données et les besoins des utilisateurs. L'utilisation des besoins utilisateurs est actuellement reconnue comme une étape indispensable pour la réussite des projets d'entreposage. L'utilisation des besoins se fait sentir à différentes phases du cycle de vie de l'entrepôt (optimisation, personnalisation, recommandations, traçabilité, etc.). Nous remarquons cependant qu'aucune trace du modèle conceptuel ni du modèle des besoins n'est sauvegardée dans l'entrepôt final. Pour remédier à ces limites, nous proposons dans cet article une piste de réflexion sur la proposition d'une nouvelle structure

de stockage d'entrepôt permettant de représenter d'une manière persistante ces trois modèles : le modèle conceptuel, le modèle des besoins et le modèle logique.
Objectif de l'article : la contribution de cet article est de présenter une architecture d'entrepôt faisant cohabiter les modèles de données (conceptuel et logique) et le modèle de besoins.

- Selma KHOURI, Ladjel BELLATRECHE, "Valorisation des besoins des utilisateurs dans le processus de conception et d'exploitation des entrepôts de données, 4èmes Journées Francophones sur les Ontologies (JFO'2011), 2011. [99].

Résumé. La conception d'un entrepôt de données (ED) repose sur les sources de données et sur les besoins des utilisateurs. Les besoins des utilisateurs n'ont pas toujours été considérés. Il est actuellement reconnu par plusieurs études que la conception des EDs doit inclure une première phase d'analyse des besoins. L'utilisation des besoins des utilisateurs se fait sentir à différentes étapes d'exploitation de l'entrepôt (optimisation, personnalisation, recommandation, traçabilité, etc). Nous remarquons cependant qu'aucune représentation des besoins n'est sauvegardée une fois la phase d'analyse des besoins terminée. Nous estimons qu'il est important d'avoir une vue persistante des besoins au sein de l'entrepôt, au même titre que les données des sources. Nous proposons une méthode de conception d'un ED répondant à cet objectif. Nous formalisons le modèle des besoins en exploitant une ontologie de domaine. L'ontologie permet de représenter un schéma intégré des sources étendu par le modèle des besoins, et sera stocké dans l'entrepôt final.
Objectif de l'article : la contribution de cet article accepté comme poster est de présenter la méthode de conception fournissant le mécanisme de persistance des besoins des utilisateurs au sein de l'ED.

- Selma KHOURI, Ladjel BELLATRECHE, Patrick Marcel, Embedding User's Requirements in Ontology-based Data Warehouses, International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE), October, 2011. [103].

Résumé. User requirements collection and analysis have a strategic significance in the design of databases and data warehouses. Historically, they were not well exploited in the initial studies on data warehouse design. Afterwards, they were recognized as a major factor in determining the success of the warehouse project due to their strong impact on the definition of traditional tasks of data warehouse life cycle : conceptual model, physical model, maintenance, etc. Their usefulness goes beyond design stages and covers many exploitation and analysis phases such as optimization, personalization and data mining. In the current situation, we notice that once the data warehouse is designed, no tracking of requirements is kept in its model. In this paper, we first study the impact of making user requirements *persistent* within the data warehouse repository on its construction and exploitation. We then model user requirements expressed by user-goal formalism. This model is then connected to an ontology-based design method. Finally, a case tool supporting our proposal is presented.
Objectif de l'article : la contribution de cet article accepté comme poster est de présenter la méthode de conception fournissant le mécanisme de persistance des besoins des utilisateurs au sein de l'ED, une étude théorique démontrant l'impact de la persistance des besoins au sein de la structure de l'ED et une validation de la faisabilité de la persistance des besoins dans une

architecture d'ED sémantique. La méthode complète la méthode présentée dans les articles précédents fournissant le schéma de l'ED.

- Selma KHOURI, Ladjel BELLATRECHE, Patrick Marcel, Towards a Method for Persisting Requirements and Conceptual Models in Data Warehousing Context, 27èmes journées de Bases de Données Avancées (BDA'2011), 2011. [171].

Résumé. User requirements collection and analysis have a strategic significance in the design of databases and data warehouses. Historically, they were not well exploited in the initial studies on data warehouse design. Afterwards, they were recognized as a major factor in determining the success of the warehouse project due to their strong impact on the definition of traditional tasks of data warehouse life cycle : conceptual model, physical model, maintenance, etc. Their usefulness goes beyond design stages and covers many exploitation and analysis phases such as optimization, personalization and data mining. In the current situation, we notice that once the data warehouse is designed, no tracking of requirements is kept in its model. In this paper, we first study the impact of making user requirements *persistent* within the data warehouse repository on its construction and exploitation. We then model user requirements expressed by user-goal formalism. This model is then connected to an ontology-based design method. Finally, a case tool supporting our proposal is presented.
Objectif de l'article : l'article porte sur la même contribution que l'article précédent accepté comme papier long.

- Ladjel BELLATRECHE, Selma KHOURI, Ilyes BOUKHARI, Rima BOUCHAKRI, Using Ontologies and Requirements for Constructing and Optimizing Data Warehouses, Proceedings of the 30th IEEE International Convention MIPRO, Opatija, Croatia, May, 2012. [19].

Résumé. Developing database (DB) and data warehouse (DW) applications pass through three main phrases imposed by the ANSI/SPARC architecture : conceptual modeling, logical modeling and physical modeling. Some research efforts add a new ontological level above the conceptual one. This architecture has created two main actors whose presence is mandatory to ensure the success of applications : "conceptual designer" for conceptual and logical levels and "database administrators" (DBA) for physical level. Note that some administration tasks need some inputs from conceptual phase. Unfortunately, interaction between these two actors is negligible. Recently, some research and industrial efforts identify a highest cost of DBA and propose tools (advisors) to replace them, in order to ensure what we call zero-administration. The main limitation of these tools is their robustness. In this paper, we propose a new human resource management for database applications. Instead of replacing DBA, we claim to delegate some DBA tasks to conceptual designers. These tasks are usually those having inputs user requirements that may be translated to SQL queries. First, we propose a user make user requirements persistent into DWs. Then, a selection of indexes based on user requirements is presented and evaluated using star schema benchmark.
Objectif de l'article : la contribution de cet article est de montrer l'impact de la persistance des besoins pour la simulation du cycle de conception de l'ED, dans le cas d'une conception ontologique dirigée par une ontologie de domaine. L'article présente une évaluation permettant de

simuler la tâche d'optimisation dès le début du projet d'entreposage (dès la phase conceptuelle) en exploitant les besoins des utilisateurs. La deuxième contribution de l'article est de présenter un nouveau modèle économique reposant sur un seul acteur : le concepteur de l'ED.

- Selma KHOURI, Ladjel BELLATRECHE, Ilyes BOUKHARI, Selma Bouarar, More Investment in Conceptual Designers : Think about it !, 15th IEEE International Conference on Computational Science and Engineering., 2012. [102].

Résumé. Developing database (DB) and data warehouse (DW) applications pass through three main phrases imposed by the ANSI/SPARC architecture : conceptual modeling, logical modeling and physical modeling. Some research efforts add a new ontological level above the conceptual one. This architecture has created two main actors whose presence is mandatory to ensure the success of applications : "conceptual designer" for conceptual and logical levels and "database administrators" (DBA) for physical level. Note that some administration tasks need some inputs from conceptual phase. Unfortunately, interaction between these two actors is negligible. Recently, some research and industrial efforts identify a highest cost of DBA and propose tools (advisors) to replace them, in order to ensure what we call zero-administration. The main limitation of these tools is their robustness. In this paper, we propose a new human resource management for database applications. Instead of replacing DBA, we claim to delegate some DBA tasks to conceptual designers. These tasks are usually those having inputs user requirements that may be translated to SQL queries. First, we propose a user make user requirements persistent into DWs. Then, a selection of indexes based on user requirements is presented and evaluated using star schema benchmark.

Objectif de l'article : cet article généralise la contribution précédente et la positionne dans le cas d'une ontologique classique, s'appliquant aussi bien pour les BD que pour les ED.

La première proposition de cette thèse portant sur la "persistance des besoins" a fait l'objet de deux publications. Une publication dans une revue francophone [104].

- Selma KHOURI, Ladjel BELLATRECHE, Patrick Marcel, Une démarche de conception d'un Entrepôt Sémantique Matérialisant les Données et les Besoins, Ingénierie des Systèmes d'Information, 2012.

Cette publication est issue de l'article [105], suite à la sélection des meilleurs articles proposés et leur réévaluation. La deuxième publication est pubiée dans une revue internationale.

- Selma KHOURI, Ilyes BOUKHARI, Ladjel BELLATRECHE, Stéphane JEAN, Eric SARDET, Mickael BARON, Ontology-based structured web data warehouses for sustainable interoperability : requirement modeling, design methodology and tool, Computers in Industry Journal, Elsevier (Factor Impact : 1.529), 2012.

Liste des articles relatifs à la deuxième proposition

- Selma KHOURI, Ladjel BELLATRECHE, Nabila Berkani, Generic Conceptual Framework for Handling Schema Diversity in Data Integration : Applications to Semantic Databases, 16th East-European Conference on Advances in Databases and Information Systems (ADBIS), edited by Springer Verlag, 2012. [100].

Résumé. DataBase Integration Systems (*DIS*) aim at providing a unified view of data stored in different *heterogeneous* local sources through a global schema. The schemas of global and local sources are represented by *a-priori known* logical representations. This makes the potential deployment model of the DIS, like for Data Warehouse (*DW*) systems, *rigid* and *inflexible*. To overcome these limitations, we claim that the integration process should be completely performed at the *conceptual level* independently of any implementation constraint. After studying existing database (*DB*) integration systems, we propose through this paper a generic conceptual framework for *DB* schema integration. This framework is generic as it *subsumes* most important *DB* integration systems studied. It is defined based on the description logic formalism. We then instantiate it through a case study considering a set of Oracle semantic *DB*, that are *DB* storing their own conceptual model, generated using Lehigh University BenchMark (LUBM). These *DB* participate in the construction of a semantic *DW*.
Objectif de l'article : la contribution de cet article est de présenter un Framework d'intégration <G,S,M> conceptuel, et générique par rapport aux systèmes d'intégration de données existants.

- Selma KHOURI, Nabila Berkani, Ladjel BELLATRECHE, Generic Methodology for Semantic DataWarehouse Design : From Schema Definition to ETL, To appears in 4-th International Conference on. Intelligent Networking and Collaborative Systems (INCoS), edited by IEEE, 2012. [23].

Résumé. Actually, any company needs to collaborate with others to improve their performance and productivity. Ontologies can provide a way to promote collaboration between companies. They contribute on reducing the syntax and semantic conflicts that may occur during the collaboration process. Data warehouse technology is a serious candidate for data-sharing architecture that may be employed within the collaborating companies. The spectacular adoption of domain ontologies by several communities facilitates the explosion of semantic databases sources ($\mathcal{SDB}$) that become candidate for building the semantic data warehouses ($\mathcal{SDW}$). This situation motivates us to deeply formalise the structure of semantic sources in order to propose an automatic construction of a semantic data warehouse $\mathcal{SDW}$. In this paper, we first proposed a generic framework for handling semantic sources. Secondly, the generic ETL steps are incorporated to our framework. Our proposal is validated through a case study ; considering *Oracle* $\mathcal{SDB}$, where each source references a global ontology of the Lehigh University BenchMark.
Objectif de l'article : la contribution de cet article est de proposer un algorithme ETL complètement conceptuel, basé sur le Framework présenté dans l'article précédent.

- Selma KHOURI, Ladjel BELLATRECHE, Nabila Berkani, MODETL : A complete MODeling and ETL method for designing Data Warehouses from Semantic Databases, International Conference on Management of Data (COMAD), 2012. [101].

Résumé. In last decades, Semantic DataBases ($\mathcal{SDB}$) emerge and the major vendors provide semantic support in their products. This is mainly due to the spectacular development of ontologies in several important domains like E-commerce, Engineering, Medicine, etc. Contrary to traditional databases, where tuples are stored in a relational (table) layout, $\mathcal{SDB}$ store ontological data according to one of three main storage layouts (horizontal, vertical, binary). In the other hand, ontologies represent a natural continuity of the conceptual models. Note that $\mathcal{SDB}$ become serious candidates for business intelligence projects built around the Data Warehouse

($\mathcal{DW}$) technology. The important steps of the life-cycle warehouse design (user requirement analysis, conceptual design, logical design, ETL, physical design) are usually dealt in isolation way. This treatment is mainly due to the complexity of each phase. Actually, $\mathcal{DW}$ technology is quite mature for traditional data sources. As a consequence, leveraging its steps to deal with semantic $\mathcal{DW}$ becomes a necessity. In this paper, we propose to cover the most important steps of life-cycle of semantic $\mathcal{DW}$. Firstly, a mathematical formalization of ontologies, $\mathcal{SDB}$ and semantic $\mathcal{DW}$ is given. User requirements are expressed on the ontological level by the means of the goal oriented paradigm. Secondly, the ETL process is expressed on the ontological level, independently of any implementation constraint. Thirdly, different deployment solutions according to the storage layouts are proposed and implemented using data access object design patterns. Finally, a prototype validating our proposal using the Lehigh University Benchmark ontology is given.
Objectif de l'article : la contribution de cet article accepté comme un " work in progress " est de présenter une méthode de conception de l'ED définissant le processus ETL au niveau de la phase conceptuelle.

- Ladjel BELLATRECHE, Selma KHOURI, Nabila Berkani, Semantic Data Warehouse Design : From ETL to Deployment à la Carte, 18th International Conference on Database Systems for Advanced Application (DASFAA 2013), 2013. [18].

Résumé. In last decades, semantic databases ($\mathcal{SDB}$) emerge and become operational databases, since the major vendors provide semantic supports in their products. This is mainly due to the spectacular development of ontologies in several domains like E-commerce, Engineering, Medicine, etc. Contrary to a traditional database, where its tuples are stored in a relational (table) layout, a $\mathcal{SDB}$ stores independently ontology and its instances in one of the three main storage layouts (horizontal, vertical, binary). Based on this situation, $\mathcal{SDB}$ become serious candidates for business intelligence projects built around the Data Warehouse ($\mathcal{DW}$) technology. The important steps of the $\mathcal{DW}$ development life-cycle (user requirement analysis, conceptual design, logical design, ETL, physical design) are usually dealt in isolation way. This is mainly due to the complexity of each phase. Actually, the $\mathcal{DW}$ technology is quite mature for the traditional data sources. As a consequence, leveraging its steps to deal with semantic $\mathcal{DW}$ becomes a necessity. In this paper, we propose a methodology covering the most important steps of life-cycle of semantic $\mathcal{DW}$. Firstly, a mathematical formalization of ontologies, $\mathcal{SDB}$ and semantic $\mathcal{DW}$ is given. User requirements are expressed on the ontological level by the means of the goal oriented paradigm. Secondly, the ETL process is expressed on the ontological level, independently of any implementation constraint. Thirdly, different deployment solutions according to the storage layouts are proposed and implemented using the data access object design patterns. Finally, a prototype validating our proposal using the Lehigh University Benchmark ontology is given.
Objectif de l'article : la contribution de cet article est de fournir une méthode de conception et d'alimentation des ED à partir de BDBO.

La deuxième proposition de cette thèse portant sur le "déploiement à la carte" a fait l'objet d'une publication internationale.

- Nabila Berkani, Ladjel BELLATRECHE, Selma KHOURI, Towards a Conceptualization of ETL and Physical Storage of Semantic Data Warehouses as a Service, Cluster Computing Journal (CCJ 2013), 2013. [22].

Nous avons également participé à la rédaction de trois articles au sein du laboratoire LIAS :

- Stéphane JEAN, Ladjel BELLATRECHE, Géraud Fokou, Mickael BARON, Selma KHOURI, OntoDBench : Novel Benchmarking System for Ontology-Based Databases, Ontologies, DataBases, and Applications of Semantics (ODBASE 2012), Septembre, 2012. [90].
- Bery MBAIOSSOUM, Selma KHOURI, Ladjel BELLATRECHE, Stéphane JEAN, Mickael BARON, Etude Comparative des Systèmes de Bases de Données à base Ontologiques, INFORSID, 2012. [136].
- Stéphane JEAN, Ladjel BELLATRECHE, Géraud Fokou, Mickael BARON, Selma KHOURI, OntoDBench : Ontology-based Database Benchmark, 28e journées Bases de Données Avancées (BDA), Clermont Ferrand, 2012. [91].

Glossaire

TBOX : Assertion Box
BD : Base de Données
DL : Description Logic
DBO : Données à base ontologique
ED : Entrepôt de Données
ETL : Extract-Transform-Load
GaV : Global as View
GlaV : Generalized Local as View
IB : Ingénierie des Besoins
LaV : Local as View
LIAS : Laboratoire d'Informatique et d'Automatique pour les Systèmes
LUBM : Lehigh University BenchMark
LD : Logiques de description
MCD : Modèle Conceptuel des Données
MDA : Model-Driven Architecture
MLD : Modèle Logique des Données
MOLAP : Multidimensional On-Line Analytical Processing
MPD : Modèle Physique des Données
OCC : Ontologies Conceptuelles Canoniques
OCNC : Ontologies Conceptuelles Non Canoniques
OL : Ontologies Linguistiques
OLAP : On-Line Analytical Processing
OWL : Ontology Web Language
PLIB : Parts Library
RDF : Ressource Description Framework
RDFS : RDF Schéma
ROLAP : Rolational On-Line Analytical Processing
SGBD : Système de Gestion de Bases de Données
SGBDR : SGBD Relationnel
SID : Système d'Intégration de Données
SQL : Structured Query Language
SSB : Star Schema Benchmark
TBOX : Terminological Box
XML : eXtensible Markup Language

Bibliographie

[1] SWAD-Europe Deliverable 10.2 : Mapping Semantic Web Data with RDBMSes.

[2] D. Abadi, A. Marcus, S. Madden, and K. Hollenbach. Sw-store : a vertically partitioned dbms for semantic web data management. *VLDB Journal*, 18(2) :385–406, 2009.

[3] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007.

[4] A. Abelló, J. Samos, and F. Saltor. A framework for the classification and description of multidimensional data models. In *Database and Expert Systems Applications*, pages 668–677. Springer, 2001.

[5] A. Abelló, J. Samos, and F. Saltor. Yam2 : a multidimensional conceptual model extending uml. *Information Systems*, 31(6) :541–567, 2006.

[6] C. Adamson. *Mastering data warehouse aggregates : solutions for star schema performance*. Wiley, 2012.

[7] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 506–521, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[8] R. e. a. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, J. Gehrke, L. Gruenwald, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. F. Korth, D. Kossmann, S. Madden, R. Magoulas, B. C. Ooi, T. O'Reilly, R. Ramakrishnan, S. Sarawagi, M. Stonebraker, A. S. Szalay, and G. Weikum. The claremont report on database research. *SIGMOD Rec.*, 37(3) :9–19, Sept. 2008.

[9] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ics-forth rdfsuite : Managing voluminous rdf description bases. In *Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb 2001)*, 2001.

[10] J.-N. M. Andrea Carme and S. Rizzi. A model-driven heuristic approach for detecting multidimensional facts in relational data sources. In *Proceedings of the 12th international conference on Data warehousing and knowledge discovery*, DaWaK'10, pages 13–24, Berlin, Heidelberg, 2010. Springer-Verlag.

[11] A. Aybuke and W. Claes. *Engineering and Managing Software Requirements*. Springer, 2005.

[12] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[13] C. Ballard, D. Herreman, D. Schau, R. Bell, E. Kim, and A. Valencic. *Data modeling techniques for data warehousing*. IBM, 1998.

[14] L. Bellatreche. *Utilisation des vues matérialisées, des index et de la fragmentation dans la conception logique et physique d'un entrepôt de données*. Thèse, Université Blaise Pascal - Clermont Ferrand, 2000.

[15] L. Bellatreche, K. Boukhalfa, and P. Richard. Referential horizontal partitioning selection problem in data warehouses : Hardness study and selection algorithms. *International Journal of Data Warehousing and Mining*, 5(4) :1–23, 2009.

[16] L. Bellatreche, N. X. Dung, G. Pierra, and D. Hondjack. Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry*, 57(8) :711–724, 2006.

[17] L. Bellatreche, A. Giacometti, A. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for olap queries. In *In Proc. of DOLAP05*, pages 9–18, 2005.

[18] L. Bellatreche, S. Khouri, and N. Berkani. Semantic data warehouse design : From etl to deployment a la carte. In *To Appear in the 18th International Conference on Database Systems for Advanced Applications (DASFAA)*, 2013.

[19] L. Bellatreche, S. Khouri, I. Boukhari, and R. Bouchakri. Using ontologies and requirements for constructing and optimizing data warehouses. In *MiproBIS*, pages 1568 – 1573, 2012.

[20] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. The momis approach to information integration. 2001.

[21] S. Bergamaschi, F. Guerra, M. Orsini, C. Sartori, and M. Vincini. A semantic approach to etl technologies. *Data & Knowledge Engineering*, 70(8) :717–731, 2011.

[22] N. Berkani, L. Bellatreche, and S. Khouri. Towards a conceptualization of etl and physical storage of semantic data warehouses as a service. *To appear in Cluster Computing : The Journal of Networks, Software Tools and Applications, Springer*, 2013.

[23] N. Berkani, S. Khouri, and L. Bellatreche. Generic methodology for semantic datawarehouse design : From schema definition to etl. In *IEEE InCos proceedings*, pages 404–411, 2012.

[24] M. Boehnlein and A. Ulbrich-vom Ende. Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems. In *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*, pages 15–21. ACM, 1999.

[25] A. Bonifati, F. Cattaneo, S. Ceri, A. Fuggetta, and S. Paraboschi. Designing data marts for data warehouses. *ACM Transactions on SEM*, 10(4) :452–483, 2001.

[26] A. Borgida and L. Serafini. Distributed description logics : Directed domain correspondences in federated information sources. In *On the Move to Meaningful Internet Systems 2002 : CoopIS, DOA, and ODBASE*, pages 36–53. Springer, 2002.

[27] R. Bouchakri and L. Bellatreche. On simplifying integrated physical database design. In *15th International Conference on Advances in Databases and Information Systems (ADBIS)*, pages 333–346, 2011.

[28] J. Broekstra, A. Kampman, and F. v. Harmelen. Sesame : A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 54–68, London, UK, UK, 2002. Springer-Verlag.

[29] R. Bruckner, B. List, and J. Schiefer. Developing requirements for data warehouse systems with use cases. In *Proc. 7th Americas Conf. on Information Systems*, pages 329–335, 2001.

[30] J. Bubenko, C. Rolland, P. Loucopoulos, and V. DeAntonellis. Facilitating fuzzy to formal requirements modelling. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 154–157. IEEE, 1994.

[31] F. S. Bulos D. Getting started with adapt tm olap database design, white paper.

[32] L. Burmester and M. Goeken. Method for user oriented modelling of data warehouse systems. In *ICEIS (3)*, pages 366–374, 2006.

[33] L. Cabibbo and R. Torlone. A logical approach to multidimensional databases. *Advances in Database Technology-EDBT'98*, pages 183–197, 1998.

[34] C. Calero, F. Ruiz, and M. Piattini. *Ontologies for Software Engineering and Software Technology*. Springer Verlag, 2006.

[35] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *The Emerging Semantic Web-Selected Papers from the First Semantic Web Working Symposium*, pages 201–214, 2002.

[36] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A principled approach to data integration and reconciliation in data warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, 1999.

[37] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Data integration in data warehousing. *International Journal of Cooperative Information Systems*, 10(03) :237–271, 2001.

[38] D. Calvanese, G. Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, and

M. Ruzzi. Data integration throughdl-lite$_a$ ontologies. In *SDKB*, pages 26–47, 2008.

[39] D. Calvanese, G. Giacomo, D. Lembo, R. Lenzerini, Rosati, and M. Ruzzi. Using owl in data integration. In *Semantic Web Information Management*, pages 397–424. 2009.

[40] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263, 1998.

[41] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Int. J. Cooperative Inf. Syst.*, 2(4) :375–398, 1993.

[42] J. M. Cavero, M. Piattini, and E. Marcos. Midea : A multidimensional data warehouse methodology. *ICEIS (1)*, pages 138–144, 2001.

[43] B. Chaib-draa. Causal maps : theory, implementation, and practical applications in multiagent environments. *IEEE TKDE*, 14(6) :1201–1217, 2002.

[44] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The tsimmis project : Integration of heterogenous information sources. 1994.

[45] P. Chen. Software pioneers. chapter Entity-relationship modeling : historical events, future trends, and lessons learned, pages 296–310. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[46] P. P.-S. Chen. The entity-relationship model toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1) :9–36, 1976.

[47] E. Codd, S. Codd, and C. Salley. Providing olap (on-line analytical processing). 1993.

[48] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6) :377–387, 1970.

[49] E. F. Codd. Data models in database management. In *ACM SIGMOD Record*, volume 11, pages 112–114. ACM, 1980.

[50] E. F. Codd. Relational database : a practical foundation for productivity. *Communications of the ACM*, 25(2) :109–117, 1982.

[51] B. Cuenca Grau, B. Parsia, and E. Sirin. Combining owl ontologies using-connections. *Web Semantics : Science, Services and Agents on the World Wide Web*, 4(1) :40–59, 2006.

[52] A. Dardenne, A. Van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1) :3–50, 1993.

[53] Z. El Akkaoui, J.-N. Mazón, A. Vaisman, and E. Zimányi. Bpmn-based conceptual modeling of etl processes. *Data Warehousing and Knowledge Discovery*, pages 1–14, 2012.

[54] K. ElGebaly and A. Aboulnaga. Robustness in automatic physical database design. In *EDBT*, pages 145–156, 2008.

[55] R. Elmasri. *Fundamentals of database systems*. Pearson Education India, 2008.

[56] C. Fankam. *OntoDB2 : un systeme flexible et efficient de Base de Données à Base Ontologique pour le Web sémantique et les données techniques*. PhD thesis, ENSMA, Decembre 2009.

[57] C. Fankam, L. Bellatreche, D. Hondjack, Y. A. Ameur, and G. Pierra. Sisro, conception de bases de données à partir d'ontologies de domaine. *Technique et Science Informatiques*, 28(10) :1233–1261, 2009.

[58] C. Fankam, S. Jean, G. Pierra, L. Bellatreche, and Y. A. Ameur. Towards connecting database applications to ontologies. In *First International Conference on Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA'09.*, pages 131–137. IEEE, 2009.

[59] J. Feng, Q. Fang, and H. Ding. Prefixcube : prefix-sharing condensed data cube. In *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, pages 38–47. ACM, 2004.

[60] E. Franconi and A. Kamblet. A data warehouse conceptual data model. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 435–436. IEEE, 2004.

[61] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax : an extensible data cleaning tool. *ACM SIGMOD Record*, 29(2) :590, 2000.

[62] I. Gam, C. Salinesi, et al. A requirement-driven approach for designing data warehouses. *Requirements Engineering : Foundation for Software Quality (REFSQ)*, 2006.

[63] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster : An information integration system. In *ACM SIGMOD Record*, volume 26, pages 539–542. ACM, 1997.

[64] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Formal reasoning tech-

niques for goal models. *Journal on Data Semantics I*, pages 1–20, 2003.

[65] P. Giorgini, S. Rizzi, and M. Garzetti. Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 47–56. ACM, 2005.

[66] M. Glinz. On non-functional requirements. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 21–26. IEEE, 2007.

[67] C. Goh, S. Bressan, E. Madnick, and M. Siegel. Context interchange : New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3) :270–293, 1999.

[68] M. Golfarelli. Data warehouse life-cycle and design. In *Encyclopedia of Database Systems*, pages 658–664. Springer US, 2009.

[69] M. Golfarelli. From user requirements to conceptual design in data warehouse design a survey. *Data Warehousing Design and Advanced Engineering Applications Methods for Complex Construction*, pages 1–16, 2010.

[70] M. Golfarelli, D. Maio, and S. Rizzi. The dimensional fact model : a conceptual model for data warehouses. *International Journal of Cooperative Information Systems*, 7(02n03) :215–247, 1998.

[71] M. Golfarelli, S. Rizzi, and P. Biondi. myolap : An approach to express and evaluate olap preferences. *Knowledge and Data Engineering, IEEE Transactions on*, 23(7) :1050–1064, 2011.

[72] M. Golfarelli, S. Rizzi, and E. Turricchia. Modern software engineering methodologies meet data warehouse design : 4wd. In *Proceedings of the 13th international conference on Data warehousing and knowledge discovery*, DaWaK'11, pages 66–79, Berlin, Heidelberg, 2011. Springer-Verlag.

[73] G.-L. G. Gómez. *Construction automatisée de l'ontologie des systèmes médiateurs : application à des systèmes intégrant des services standards accessibles via le Web*. PhD thesis, 2005.

[74] O. Grabova, J. Darmont, J.-H. Chauchat, and I. Zolotaryova. Business intelligence for small and middle-sized entreprises. *SIGMOD Record*, 39(2) :39–50, 2010.

[75] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies : Theory and practice. *Journal of Artificial Intelligence Research*, 31(1) :273–318, 2008.

[76] J. GRAY. Database systems : A textbook case of research paying off. *Committee on the Fundamentals of Computer Science : Challenges and Opportunities, editors, Computer Science : Reflections on the Field, Reflections from the Field*, pages 80–88, 1995.

[77] S. J. Greenspan. Requirements modeling : a knowledge representation approach to software requirements definition. 1984.

[78] T. Gruber. A translation approach to portable ontology specifications. *In Knowledge Acquisition*, 5(2) :199–220, 1993.

[79] P. Haase and B. Motik. A mapping system for the integration of owl-dl ontologies. In *Proceedings of the first international workshop on Interoperability of heterogeneous information systems*, pages 9–16. ACM, 2005.

[80] M.-S. Hacid and U. Sattler. An object-centered multi-dimensional data model with hierarchically structured dimensions. In *Knowledge and Data Engineering Exchange Workshop, 1997. Proceedings*, pages 65–72. IEEE, 1997.

[81] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The stanford data warehousing project. 1995.

[82] M. Hammer and D. Mc Leod. Database description with sdm : a semantic database model. *ACM Transactions on Database Systems (TODS)*, 6(3) :351–386, 1981.

[83] W. Heaven and A. Finkelstein. Uml profile to support requirements engineering with kaos. *IEE Proceedings-Software*, 151(1) :10–27, 2004.

[84] J. Heflin, J. Hendler, and S. Luke. SHOE : A Knowledge Representation Language for Internet Applications. Technical report, University of Maryland, 1999.

[85] J. M. Hellerstein and M. Stonebraker. *Readings in database systems*. MIT Press, 2005.

[86] J. M. Hellerstein, M. Stonebraker, and R. Caccia. Independent, open enterprise data integration. *IEEE Data Engineering Bulletin*, 22(1) :43–49, 1999.

[87] D. Hondjack, G. Pierra, and L. Bellatreche. Ontodb : An ontology-based database for data intensive applications. In *DASFAA*, pages 497–508, 2007.

[88] B. Husemann, J. Lechtenbörger, and G. Vossen. Conceptual data warehouse design. In *In*

Proc. of the International Workshop on Design and Management of Data Warehouses (DMDW 2000, pages 3–9, 2000.

[89] W. H. Inmon. *Building the data warehouse*. J. Wiley, 2002.

[90] S. Jean, L. Bellatreche, G. Fokou, M. Baron, and S. Khouri. Ontodbench : Novel benchmarking system for ontology-based databases. In *OTM Conferences (2)*, pages 897–914, 2012.

[91] S. Jean, L. Bellatreche, G. Fokou, M. Baron, and S. Khouri. Ontodbench : Ontology-based database benchmark. In *28e journées Bases de Données Avancées (BDA)*, 2012.

[92] M. Jensen, T. Holmgren, and T. Pedersen. Discovering multidimensional structure in relational data. *Data Warehousing and Knowledge Discovery*, pages 138–148, 2004.

[93] M. A. Jeusfeld, C. Quix, and M. Jarke. Design and analysis of quality information for data warehouses. In *Conceptual Modeling ER'98*, pages 349–362. Springer, 1998.

[94] E. Jiménez-Ruiz, R. Berlanga, V. Nebot, and I. Sanz. Ontopath : A language for retrieving ontology fragments. In *On the Move to Meaningful Internet Systems 2007 : CoopIS, DOA, ODBASE, GADA, and IS*, pages 897–914. Springer, 2007.

[95] E. Jiménez-Ruiz, B. Grau, U. Sattler, T. Schneider, and R. Berlanga Llavori. Safe and economic re-use of ontologies : A logic-based methodology and tool support. In *Description Logics*, 2008.

[96] S. Khouri and L. Bellatreche. Ontostar : Outil de conception multidimensionnelle d'un entrepôt de données à base ontologique. In *26èmes journées Bases de Données Avancées (BDA'2010)*, 2010.

[97] S. Khouri and L. Bellatreche. Dwobs : data warehouse design from ontology-based sources. In *Database Systems for Advanced Applications*, pages 438–441. Springer, 2011.

[98] S. Khouri and L. Bellatreche. Osons tout persister dans un entrepôt : données et modèles. In *EDA'11*, pages 39–53, 2011.

[99] S. Khouri and L. Bellatreche. Valorisation des besoins des utilisateurs dans le processus de conception et d'exploitation des entrepots de données. In *4èmes Journées Francophones sur les Ontologies (JFO'2011)*, 2011.

[100] S. Khouri, L. Bellatreche, and N. Berkani. Generic conceptual framework for handling schema diversity during source integration. In *ADBIS (2)*, pages 149–160, 2012.

[101] S. Khouri, L. Bellatreche, and N. Berkani. Modetl : A complete modeling and etl method for designing data warehouses from semantic databases. In *International Conference on Management of Data (COMAD)*, 2012.

[102] S. Khouri, L. Bellatreche, I. Boukhari, and S. Bouarar. More investment in conceptual designers : Think about it ! In *CSE'12*, pages 88–93, 2012.

[103] S. Khouri, L. Bellatreche, and P. Marcel. Embedding user's requirements in data warehouse repositories. In *OTM Workshops, Springer LNCS*, pages 35–36, 2011.

[104] S. Khouri, L. Bellatreche, and P. Marcel. Une démarche de conception d'un entrepôt sémantique matérialisant les données et les besoins. *Ingénierie des Systèmes d'Information*, 17(5) :9–34, 2012.

[105] S. Khouri, I. Boukhari, L. Bellatreche, S. Jean, E. Sardet, and M. Baron. Ontology-based structured web data warehouses for sustainable interoperability : requirement modeling, design methodology and tool. *Computers in Industry*, pages 799–812, 2012.

[106] S. Khouri and B. Ladjel. A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In *Proceedings of the ACM 13th international workshop on Data warehousing and OLAP*, pages 19–24. ACM, 2010.

[107] R. Kimball. *The data warehouse toolkit : practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[108] R. Kimball, L. Reeves, W. Thornthwaite, M. Ross, and W. Thornwaite. *The Data Warehouse Lifecycle Toolkit : Expert Methods for Designing, Developing and Deploying Data Warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998.

[109] R. Kimball, M. Ross, et al. The data warehouse toolkit : the complete guide to dimensional modelling. *New York ua*, 2002.

[110] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. B. Zdonik. Coradd : Correlation aware database designer for materialized views and indexes. *Proceedings of the VLDB Endowment*, 3(1-2) :1103–1113, 2010.

[111] A. Kiryakov, D. Ognyanov, and D. Manov. Owlim–a pragmatic semantic repository for owl. In *Web Information Systems Engineering–WISE 2005 Workshops*, pages 182–192. Springer, 2005.

[112] T. Kraska and U. Röhm. Genea : Schema-aware mapping of ontologies into relational databases. In *COMAD*, pages 92–103. Tata McGraw-Hill Publishing Company Limited, 2006.

[113] G. E. Krasner, S. T. Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3) :26–49, 1988.

[114] W. J. Labio, J. L. Wiener, H. Garcia-Molina, and V. Gorelik. Efficient resumption of interrupted warehouse loads. *SIGMOD Rec.*, 29(2) :46–57, May 2000.

[115] L. V. Lakshmanan, J. Pei, and Y. Zhao. Qc-trees : An efficient summary structure for semantic olap. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 64–75. ACM, 2003.

[116] J. Lechtenbörger. *Data warehouse schema design*, volume 79. IOS Press, 2001.

[117] J. Lechtenbörger and G. Vossen. Multidimensional normal forms for data warehouse design. *Information Systems*, 28(5) :415–434, 2003.

[118] W. Lehner, J. Albrecht, and H. Wedekind. Normal forms for multidimensional databases. In *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*, pages 63–72. IEEE, 1998.

[119] H.-J. Lenz and A. Shoshani. Summarizability in olap and statistical data bases. In *Scientific and Statistical Database Management, 1997. Proceedings., Ninth International Conference on*, pages 132–143. IEEE, 1997.

[120] M. Lenzerini. Data integration : A theoretical perspective. In *PODS*, pages 233–246, 2002.

[121] M. Levene and G. Loizou. Why is the snowflake schema a good data warehouse design ? *Information Systems*, 28(3) :225–240, 2003.

[122] B. List, J. Schiefer, and A. Tjoa. Process-oriented requirement analysis supporting the data warehouse design process a use case driven approach. In *Database and Expert Systems Applications*, pages 593–603. Springer, 2000.

[123] O. López, M. A. Laguna, and F. J. García. Metamodeling for requirements reuse. In *Anais do WER02-Workshop em Engenharia de Requisitos, Valencia, Spain*, 2002.

[124] J. Lu, L. Ma, L. Zhang, J.-S. Brunner, C. Wang, Y. Pan, and Y. Yu. Sor : a practical system for ontology storage, reasoning and search. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1402–1405. VLDB Endowment, 2007.

[125] S. Luján-Mora and J. Trujillo. Physical modeling of data warehouses using uml. In *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, DOLAP '04, pages 48–57, New York, NY, USA, 2004. ACM.

[126] S. Luján-Mora, J. Trujillo, and I.-Y. Song. A uml profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering*, 59(3) :725–769, 2006.

[127] S. Luján-Mora, P. Vassiliadis, and J. Trujillo. Data mapping diagrams for data warehouse design with uml. *Conceptual Modeling ER 2004*, pages 191–204, 2004.

[128] L. Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu. Rstar : an rdf storage and query system for enterprise resource management. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 484–491, New York, NY, USA, 2004. ACM.

[129] E. Malinowski and E. Zimányi. Hierarchies in a multidimensional model : From conceptual modeling to logical representation. *Data & Knowledge Engineering*, 59(2) :348–377, 2006.

[130] A. Maté and J. Trujillo. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. In *Advanced Information Systems Engineering*, pages 123–137. Springer, 2011.

[131] A. Maté and J. Trujillo. A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses. *Information Systems*, 2012.

[132] J.-N. Mazón, J. Lechtenbörger, and J. Trujillo. A survey on summarizability issues

in multidimensional modeling. *Data Knowl. Eng.*, 68(12) :1452–1469, 2009.

[133] J.-N. Mazón and J. Trujillo. An mda approach for the development of data warehouses. *Decision Support Systems*, 45(1) :41–58, 2008.

[134] J.-N. Mazón, J. Trujillo, and J. Lechtenbörger. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. *Data & Knowledge Engineering*, 63(3) :725–751, 2007.

[135] J.-N. Mazon, J. Trujillo, M. Serrano, and M. Piattini. Applying mda to the development of data warehouses. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 57–66. ACM, 2005.

[136] B. Mbaiossoum, S. Khouri, L. Bellatreche, S. Jean, and M. Baron. Etude comparative des systèmes de bases de données à base ontologiques. In *INFORSID*, pages 379–394, 2012.

[137] B. McBride. Jena : Implementing the rdf model and syntax specification. 2001.

[138] D. L. Moody and M. A. Kortink. From enterprise models to dimensional models : a methodology for data warehouse and data mart design. *DMDW'00, Sweden*, 5, 2000.

[139] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos : Representing knowledge about information systems. *ACM Transactions on Information Systems (TOIS)*, 8(4) :325–362, 1990.

[140] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps : Resolving schema, tuple, and value inconsistencies. *IEEE Data Eng. Bull*, 29(2) :21–31, 2006.

[141] I. Navas-Delgado and J. F. Aldana-Montes. A design methodology for semantic web database-based systems. In *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2 - Volume 02*, ICITA '05, pages 233–237, Washington, DC, USA, 2005. IEEE Computer Society.

[142] S. B. Navathe. Evolution of data modeling for databases. *Communications of the ACM*, 35(9) :112–123, 1992.

[143] V. Nebot and R. Berlanga. Building data warehouses with semantic data. In *Proceedings of the 2010 EDBT/ICDT Workshops*, page 9. ACM, 2010.

[144] V. Nebot and R. Berlanga. Building data warehouses with semantic web data. *Decision Support Systems*, 2011.

[145] V. Nebot, R. Berlanga, J. Pérez, M. Aramburu, and T. Pedersen. Multidimensional integrated ontologies : a framework for designing semantic data warehouses. *Journal on Data Semantics XIII*, pages 1–36, 2009.

[146] D. X. NGuyen. *Intégration de bases de données hétérogènes par articulation à priori d'ontologies : application aux catalogues de composants industriels*. PhD thesis, Sciences pour l'Ingénieur et Aéronautique, Décembre 2006.

[147] T. Niemi, J. Nummenmaa, and P. Thanisch. Logical multidimensional database design for ragged and unbalanced aggregation hierarchies. In *Proceedings of the International Workshop on Design and Management of Data Warehouses. Interlaken, Switzerland*, 2001.

[148] Z. Pan and J. Heflin. Dldb : Extending relational databases to support semantic web queries. In *In PSSS*, pages 109–113, 2003.

[149] M.-J. Park, J. Lee, C.-H. Lee, J. Lin, O. Serres, and C.-W. Chung. An efficient and scalable management of ontology. In *Advances in Databases : Concepts, Systems and Applications*, pages 975–980. Springer, 2007.

[150] C. Phipps and K. C. Davis. Automating data warehouse conceptual schema design and evaluation. In *DMDW*, pages 23–32, 2002.

[151] G. Pierra. Context-explication in conceptual ontologies : the plib approach. In *Proceedings of 10th ISPE International Conference on Concurrent Engineering : Research and Applications (ce'03) : Special Track on Data Integration in Engineering*, pages 243–253, 2003.

[152] N. Prakash, Y. Singh, and A. Gosain. Informational scenarios for data warehouse requirements elicitation. *Conceptual Modeling ER 2004*, pages 205–216, 2004.

[153] N. Prat and J. Akoka. From uml to rolap multidimensional databases using a pivot model. *Pucheral*, page 24, 2002.

[154] N. Prat, J. Akoka, and I. Comyn-Wattiau. A uml-based data warehouse design method. *Decision Support Systems*, 42(3) :1449–1473, 2006.

[155] V. Psyché, O. Mendes, J. Bourdeau, et al. Apport de l'ingénierie ontologique aux environnements de formation à distance. *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, 10 :89–126, 2003.

[156] M. Rafanelli and A. Shoshani. Storm : A statistical object representation model. *Statistical and Scientific Database Management*, pages 14–29, 1990.

[157] Raman01. Potter's wheel : An interactive data cleaning system. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 381–390, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[158] S. Rizzi, A. Abelló, J. Lechtenbörger, and J. Trujillo. Research in data warehouse modeling and design : dead or alive ? In *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pages 3–10. ACM, 2006.

[159] C. Rolland. Ingénierie des Besoins : L'Approche L'Ecritoire. *Journal Techniques de l'Ingénieur*, 2003.

[160] C. Rolland. Reasoning with goals to engineer requirements. *Enterprise Information Systems V*, pages 12–20, 2005.

[161] C. Rolland and A. Flory. Conception des systèmes d'information : Etat de l'art et nouvelles perspectives. In *Actes du congrès INFORSID*, pages 3–40, 1990.

[162] O. Romero and A. Abelló. Automating multidimensional design from ontologies. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 1–8. ACM, 2007.

[163] O. Romero and A. Abelló. Automatic validation of requirements to support multidimensional design. *Data Knowl. Eng.*, 69(9) :917–942, Sept. 2010.

[164] O. Romero and A. Abelló. A framework for multidimensional design of data warehouses from ontologies. *Data Knowl. Eng.*, 69(11) :1138–1157, Nov. 2010.

[165] O. Romero, D. Calvanese, A. Abello, and M. Rodriguez-Muro. Discovering functional dependencies for multidimensional design. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP (DOLAP09)*, pages 1–8, 2009.

[166] O. Romero, A. Simitsis, and A. Abelló. Gem : requirement-driven generation of etl and multidimensional conceptual designs. *Data Warehousing and Knowledge Discovery*, pages 80–95, 2011.

[167] B. S., H. P., S. L., and S. H. Formal and conceptual comparison of ontology mapping languages. In *Modular Ontologies*, pages 267–291. Springer-Verlag, Berlin, Heidelberg, 2009.

[168] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. Extending the e/r model for the multidimensional paradigm. *Advances in Database Technologies*, pages 1947–1947, 1999.

[169] J. Schiefer, B. List, and R. Bruckner. A holistic approach for managing requirements of data warehouse systems. In *8th Americas Conference on Information Systems (AMCIS 2002)*, pages 77–87. Citeseer, 2002.

[170] J. Seidenberg and A. Rector. Web ontology segmentation : analysis, classification and use. In *Proceedings of the 15th international conference on World Wide Web*, pages 13–22. ACM, 2006.

[171] P. M. Selma KHOURI, Ladjel BELLATRECHE. Towards a method for persisting requirements and conceptual models in data warehousing context. In *27èmes journées de Bases de Données Avancées (BDA'2011)*, 2011.

[172] A. Silberschatz, M. Stonebraker, and J. Ullman. Database research : achievements and opportunities into the 1st century. *ACM sIGMOD record*, 25(1) :52–63, 1996.

[173] A. Simitsis. Modeling and managing etl processes. In *In : VLDB PhD Workshop*, 2003.

[174] A. Simitsis. Mapping conceptual to logical models for etl processes. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 67–76. ACM, 2005.

[175] A. Simitsis, D. Skoutas, and M. Castellanos. Natural language reporting for etl processes. In *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP*, pages 65–72. ACM, 2008.

[176] A. Simitsis, D. Skoutas, and M. Castellanos. Representation of conceptual etl designs in natural language using semantic web technology. *Data & Knowledge Engineering*, 69(1) :96–115, 2010.

[177] A. Simitsis and P. Vassiliadis. A methodology for the conceptual modeling of etl processes. In *CAiSE workshops*, 2003.

[178] A. Simitsis and P. Vassiliadis. A method for the mapping of conceptual designs to logical blueprints for etl processes. *Decision Support Systems*, 45(1) :22–40, 2008.

[179] Y. Sismanis and N. Roussopoulos. The polynomial complexity of fully materialized coalesced cubes. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 540–551. VLDB Endowment, 2004.

[180] D. Skoutas and A. Simitsis. Designing etl processes using semantic web technologies. In *Data Warehousing and OLAP : Proceedings of the 9 th ACM international workshop on Data warehousing and OLAP*, volume 10, pages 67–74, 2006.

[181] D. Skoutas and A. Simitsis. Ontology-based conceptual design of etl processes for both structured and semi-structured data. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(4) :1–24, 2007.

[182] J. M. Smith and D. C. Smith. Database abstractions : aggregation and generalization. *ACM Transactions on Database Systems (TODS)*, 2(2) :105–133, 1977.

[183] I. Sommerville and G. Kotonya. *Requirements Engineering : Processes and Techniques*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[184] I. Y. Song, R. Khare, and B. Dai. Samstar : a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 9–16. ACM, 2007.

[185] V. Stefanov and B. List. Business metadata for the datawarehouse - weaving enterprise goals and multidimensional models. In *EDOC Workshops*, page 20, 2006.

[186] T. Stöhr, R. Müller, and E. Rahm. An integrative and uniform model for metadata management in data warehousing environments. In *Proceedings of the International Workshop on Design and Management of Data Warehouses, Heidelberg, Germany*, volume 189, 1999.

[187] J. Stéphane. *Langage d'exploitation de base de données ontologiques*. Dea t3ia, Université de Poiters, 2004.

[188] J. Stéphane. *OntoQL, un langage d'exploitation des bases de données à base ontologique*. PhD thesis, Sciences pour l'Ingénieur et Aéronautique, 2007.

[189] V. Sugumaran and V. C. Storey. The role of domain ontologies in database design : An ontology management and conceptual modeling environment. *ACM Trans. Database Syst.*, 31(3) :1064–1094, Sept. 2006.

[190] A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini. The tropos metamodel and its use. *INFORMATICA-LJUBLJANA-*, 29(4) :401, 2005.

[191] The Standish Group. Chaos report, 1995. `http://www.cs.nmt.edu/~cs328/reading/Standish.pdf` – last visited 15^{th} of June, 2008.

[192] C. Thomsen and T. Bach Pedersen. Pygrametl : A powerful programming framework for extract-transform-load programmers. In *Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP*, DOLAP '09, pages 49–56, New York, NY, USA, 2009. ACM.

[193] J. Trujillo and S. Luján-Mora. A uml based approach for modeling etl processes in data warehouses. *Conceptual Modeling - ER 2003*, pages 307–320, 2003.

[194] N. Tryfona, F. Busborg, and J. G. Borch Christiansen. starer : a conceptual model for data warehouse design. In *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*, pages 3–8. ACM, 1999.

[195] D. Tsichritzis and A. Klug. The ansi/x3/sparc dbms framework report of the study group on database management systems. *Information systems*, 3(3) :173–191, 1978.

[196] V. Tziovara, P. Vassiliadis, and A. Simitsis. Deciding the physical implementation of etl workflows. In *Proceedings of the ACM Tenth International Workshop on Data Warehousing and OLAP*, DOLAP '07, pages 49–56, New York, NY, USA, 2007. ACM.

[197] A. Van Lamsweerde. Requirements engineering in the year 00 : A research perspective. In *Proceedings of the 22nd international conference on Software engineering*, pages 5–19. ACM, 2000.

[198] A. Van Lamsweerde. Goal-oriented requirements engineering : A guided tour. In *Requirements Engineering, 2001. Proceedings.*

Fifth IEEE International Symposium on, pages 249–262. IEEE, 2001.

[199] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. Goal question metric (gqm) approach. *Encyclopedia of Software Engineering*, 2002.

[200] P. Vassiliadis, M. Bouzeghoub, and C. Quix. Towards quality-oriented data warehouse usage and evolution. *Inf. Syst.*, 25(2) :89–115, 2000.

[201] P. Vassiliadis and A. Simitsis. Extraction, transformation, and loading. In *Encyclopedia of Database Systems*, pages 1095–1101. 2009.

[202] P. Vassiliadis, A. Simitsis, and E. Baikousi. A taxonomy of etl activities. In *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pages 25–32. ACM, 2009.

[203] P. Vassiliadis, A. Simitsis, P. Georgantas, and M. Terrovitis. A framework for the design of etl scenarios. In *Advanced Information Systems Engineering*, pages 1031–1031. Springer, 2003.

[204] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis, and S. Skiadopoulos. A generic and customizable framework for the design of etl scenarios. *Information Systems*, 30(7) :492–525, 2005.

[205] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual modeling for etl processes. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21. ACM, 2002.

[206] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Modeling etl activities as graphs. In *Proc. 4th Intl. Workshop on Design and Management of Data Warehouses (DMDW)*, pages 52–61, 2002.

[207] U. Visser, H. Stuckenschmidt, H. Wache, and T. Vögele. Enabling technologies for interoperability. In *Workshop on the 14th International Symposium of Computer Science for Environmental Protection*, pages 35–46. Citeseer, 2000.

[208] R. Volz, D. Oberle, S. Staab, and B. Motik. Kaon server - a semantic web management system. In *Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003*. ACM, 2003.

[209] B. Vrdoljak, M. Banek, and S. Rizzi. Designing web warehouses from xml schemas. *Data Warehousing and Knowledge Discovery*, pages 89–98, 2003.

[210] V. T. V. U. S. H. S. G. N. H. Wache, H. and S. Hiibner. Ontology-based integration of information - a survey of existing approaches. In *OIS*, pages 108–117, 2001.

[211] V. M. B. Werneck, A. d. P. A. Oliveira, and J. Leite. Comparing gore frameworks : i-star and kaos. In *Workshop em Engenharia de Requisitos (WER 2009), Val Paraiso, Chile*, 2009.

[212] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds, et al. Efficient rdf storage and retrieval in jena2. In *Proceedings of SWDB*, volume 3, pages 7–8, 2003.

[213] R. Winter and B. Strauch. A method for demand-driven information requirements analysis in data warehousing projects. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2003.

[214] R. Winter and B. Strauch. Information requirements engineering for data warehouse systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1359–1365. ACM, 2004.

[215] Z. Wu, G. Eadon, S. Das, E. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan. Implementing an inference engine for rdfs/owl constructs and user-defined rules in oracle. In *ICDE*, pages 1239–1248, 2008.

[216] W. Y., H. P., and B. J. A survey of formalisms for modular ontologies. In *In : IJCAI 2007. Workshop SWeCKa.*, 2007.

[217] E. Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11, 2011.

[218] K. Yue. What does it mean to say that a specification is complete ? In *Proc. IWSSD-4, Fourth International Workshop on Software Specification and Design*, 1987.

Résumé

Les ED deviennent des composants incontournables dans les entreprises et les organisations. Le thème de conception des ED a fait l'objet de recherches actives ces dernières années. La principale limitation des approches proposées est le manque d'une vision globale s'inscrivant dans le cadre du cycle de conception des ED, même si la communauté reconnait toutes les phases de ce cycle. Nos principales contributions dans cette thèse portent sur la proposition d'une méthode de conception adaptée aux récentes évolutions qu'a connu le cycle de conception, et englobant l'ensemble de ses phases. Le cycle de conception a connu une diversification importante des modèles de stockage de données et des architectures de déploiement possibles offrant des choix de conception variés. Ce cycle reconnait l'importance des besoins des utilisateurs dans le processus de conception, et l'importance d'accès et de représentation de la sémantique des données. Notre première proposition présente une méthode de conception suivant une approche à base d'ontologies de domaine, permettant de valoriser les besoins des utilisateurs en leur offrant une vue persistante au sein de l'ED. Cette vue permet d'anticiper diverses tâches de conception et de simuler les différents choix de conception. Notre deuxième proposition revisite le cycle de conception en exécutant la phase ETL (extraction-transformation-chargement des données) dès la phase conceptuelle. Cette proposition permet de fournir un moyen de déploiement multiple sur différentes plateformes disponibles.

Mots-clés: Système de stockage des données, Entrepôt de données, Cycle de conception, Ontologie, Besoins des utilisateurs, Phase ETL, Déploiement.

Abstract

Data Warehouses (DWs) become essential components for companies and organizations. DW design field has been actively researched in recent years. The main limitation of the proposed approaches is the lack of an overall vision covering the DW design cycle. Our main contribution in this thesis is to propose a method adapted to recent evolutions of the DW design cycle, and covering all its phases. These evolutions have given rise to new data storage models and new deployment architectures, which offers different design choices for designers and administrators. DW literature recognizes the importance of user requirements in the design process, and the importance of accessing and representing data semantics. We propose an ontology driven design method that valorizes users' requirements by providing them a persistent view in the DW structure. This view allows anticipating diverse design tasks and simulating different design choices. Our second proposal revisits the design cycle by executing the ETL phase (extraction-transformation-loading of data) in the conceptual stage. This proposal allows a deployment *à la carte* of the DW using the different deployment platforms available.

Keywords: Database, Data warehouse, Design cycle, Ontology, User requirements, ETL phase, Deployment